Universidad Carlos III de Madrid

Escuela Politécnica Superior

Departamento de Ingeniería de Sistemas y Automática

Proyecto Fin de Carrera

Ingeniería Técnica Industrial: Electrónica Industrial

OBTENCIÓN DE LOS PARES CREADOS POR EL BRAZO MÓVIL ARTICULADO DE MANFRED MEDIANTE SISTEMA DE ADQUISICIÓN DE DATOS

Autor:

Esteban de Torres Rivera

Tutor:

Luis Moreno Lorente

Mayo 2006

AGRADECIMIENTOS

Durante todos mis años universitarios, han sido muchas las personas que han aportado su granito de arena, ya sea grande o pequeño, ya sea moral o material; a todas esas personas, que han estado en el lugar adecuado, les agradezco su paciencia y ayuda; en especial a Sergio, Jorge y Mario**.**

No quisiera desmerecer ni olvidar a nadie de los que han tomado partida durante la realización de este proyecto, todos y cada uno de ellos han sido muy importantes, pues sin su ayuda, no hubiese sido posible la realización del mismo. Doy las gracias personalmente, a Luis, mi tutor, por sus consejos; y a todas las personas que han trabajado conmigo en el laboratorio. (Nacho, Laura, Dolores...)

Quisiera terminar dando las gracias a mi madre, por su ayuda moral y sus alentadoras palabras en los momentos más difíciles. A mi padre, por su “aporte técnico”, y las interminables discusiones. A mi hermana, por las conversaciones que me han hecho recapacitar. Y a mi hermano, por las horas de piques en “la Play”. De la misma manera, doy las gracias a toda mi familia, desde mis abuelos hasta mis primos, pasando por mis tíos, por su apoyo durante todo este tiempo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN Y OBJETIVOS

1.1 INTRODUCCIÓN

En la actualidad, el vertiginoso desarrollo de la electrónica y la microelectrónica, ha motivado que todas las esferas de la vida humana se estén automatizando; por ejemplo: La industria, el hogar, los transportes, las comunicaciones, etc. En todo ese proceso de automatización, los controladores y microcontroladores juegan un papel de suma importancia.

Este desarrollo, junto con los grandes avances creados en la tecnología informática, han provocado que la forma de controlar cualquier máquina sea digital. Por este motivo ha sido necesario desarrollar unos sistemas de adquisición de datos, capaces de obtener las señales emitidas por los sensores y transformarlas en señales digitales. De esta manera el microprocesador que controla el funcionamiento, los puede interpretar y actuar de acuerdo con lo obtenido.

En las últimas décadas, todo este desarrollo ha sido muy útil para la realización de máquinas autónomas; máquinas capaces de interactuar con el entorno y modificar sus pautas de comportamiento dependiendo de él.

Manfred esta siendo diseñado para el trabajo en entornos humanos. Esto motiva que sea de vital importancia un sistema de adquisición de datos para poder controlar la fuerza que hace el brazo articulado.

1.2 OBJETIVO Y DESCRIPCIÓN DEL PROYECTO

El presente proyecto consiste en la obtención del par motor instantáneo generado por los motores de las articulaciones del manipulador ligero LWR-UC3M-1, implementado en Manfred. Esto permitirá poder controlar de forma precisa la fuerza desarrollada en cada articulación durante el movimiento. Para ello, se necesita un sistema de adquisición de datos.

Mediante un programa informático se manejará la unidad de adquisición para obtener los datos de cada motor, que serán procesados y mostrados en la pantalla del ordenador.

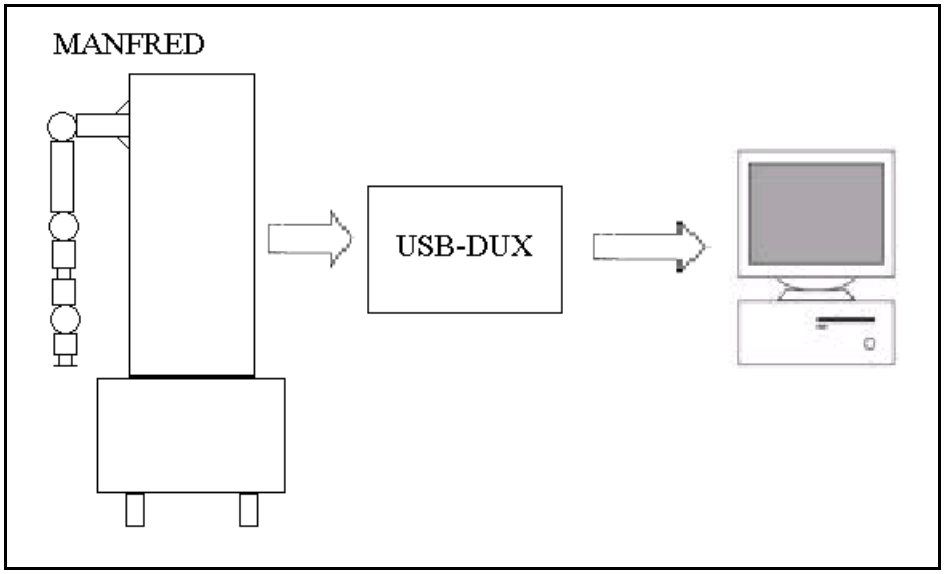

Figura 1.1: Esquema del proyecto.

En este proyecto, la unidad de adquisición de datos debía cumplir dos requisitos: Trabajar bajo sistema operativo Linux y comunicarse con el ordenador a través de un puerto USB. Estos dos requisitos nos llevan a utilizar el sistema de adquisición de datos llamado “USB-DUX”, desarrollado por la Universidad de Stirling, en el Reino Unido.

CAPÍTULO 2

MANIPULADOR MÓVIL MANFRED

2.1 DESCRIPCIÓN DE MANFRED

Manfred ha sido diseñado por el departamento de Sistemas y Automática de la Universidad Carlos III de Madrid para la realización de actividades en entornos humanos. Sus características antropomórficas le confieren la capacidad de manipulación de objetos diseñados para ser utilizados por personas. El objetivo a largo plazo es que Manfred sea capaz de funcionar de forma robusta, y al mismo tiempo, segura para las personas que lo rodean. Para este propósito, Manfred incorpora una base de tipo diferencial que le da la movilidad en distintos entornos, un brazo robot articulado que le da la capacidad de manipulación y un sistema sensorial basado en visión artificial y telemetría láser.

Todas estas características, junto con la posibilidad de un funcionamiento tanto autónomo como teleoperado, le capacitan para la realización de multitud de tareas, por ejemplo, la manipulación remota de objetos, o la realización autónoma de objetivos previamente fijados.

2.2 CARACTERÍSTICAS DE MANFRED

A continuación se describen algunas de las características de los componentes principales de Manfred.

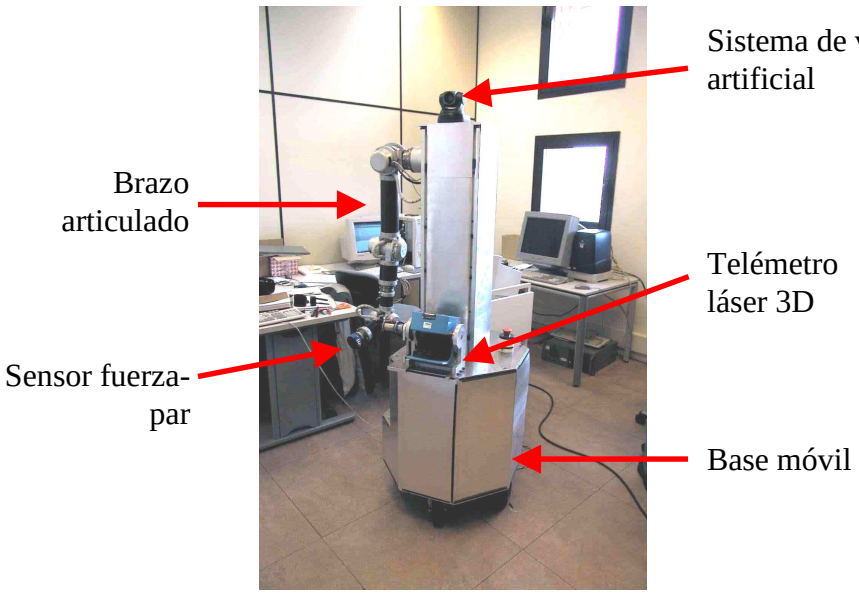

Figura 2.1: Robot móvil Manfred

Base móvil: Permite el desplazamiento dentro de los ambientes de trabajo. En su interior se encuentran los ordenadores que controlan los sistemas de visión y cinemático, además de las baterías que le dotan de autonomía. Se desplaza por medio de dos ruedas motrices bajo la base.

Telémetro láser 3D: Proporciona información en tiempo real de las distancias existentes entre Manfred y los objetos circundantes. El telémetro láser se encuentra situado encima de la base, en la parte delantera de Manfred. El elemento principal es un escáner láser bidimensional, fabricado por la casa Sick.

Sistema de visión artificial: Está formado por una cámara Sony EVI-D100 con sensor de imagen CCD. Permite a Manfred obtener imágenes de su entorno para su posterior análisis. Se encuentra ubicada en la parte superior de Manfred.

Sensor fuerza-par: Este sensor está acoplado a la muñeca del brazo. Su finalidad es medir la fuerza desarrollada en esta articulación, pudiendo así realizar tareas de manipulación basadas en el control de fuerza o par. Fabricado por la casa JR3.

Manipulador LWR-UC3M-1: Fabricado en aluminio y fibra de carbono, permite realizar las tareas de manipulación. Es un brazo robot íntegramente diseñado en la Universidad Carlos III de Madrid. Las características principales de este brazo son su bajo peso y una capacidad de carga de hasta 6 Kg. Dispone de seis grados de libertad y un alcance de 995 mm.

Este proyecto se centra única y exclusivamente en la medición del par desarrollado por cada una de las seis articulaciones de dicho brazo, analizando los motores y los reductores que las conforman.

2.3 MOTORES

Desde que se empezó a investigar en la rama de la robótica ha surgido la necesidad de utilizar motores de bajo peso, que sean rápidos, robustos y tengan poco mantenimiento.

Los motores de corriente continua convencionales son una buena alternativa, pues son fáciles de controlar, rápidos y relativamente robustos. Al poseer escobillas, el arco eléctrico que se produce al rozar éstas contra las delgas del colector, lo imposibilitan para el trabajo en entornos inflamables. Así mismo, el desgaste al que están sometidas, hace que el motor necesite mantenimiento regular.

El desarrollo de la electrónica posibilitó la utilización de controladores electrónicos para los motores. En consecuencia, los motores de corriente continua convencionales cayeron en desuso, siendo sustituidos por motores paso a paso, cuya principal característica es carecer de escobillas. Esto redujo el mantenimiento necesario.

Un tipo especial de motores paso a paso son los denominados “brushless”, o sin escobillas. A estos motores se les denomina específicamente “sin escobillas”, pues simulan el funcionamiento de un motor de corriente continua convencional sin el inconveniente de la conmutación mecánica. Una característica fundamental de esto motores, es la incorporación de unos sensores de posición en el rotor. Estos sensores, que normalmente son de efecto Hall, son utilizados para controlar la rotación del motor.

2.3.1 MOTORES PASO A PASO

Los motores paso a paso están especialmente diseñados para la construcción de mecanismos que requieren movimientos muy precisos. Su principal característica es la capacidad de moverlos un paso por cada pulso que se les aplique. Por ejemplo, un motor con un paso de 90º necesitaría 4 pulsos para completar un giro completo (360º), mientras que un motor con un paso de 1,8º necesitaría 200 pulsos. Una característica adicional es la posibilidad de quedar enclavados en una posición si una o más de sus bobinas está energizada; por el contrario, el motor quedará completamente libre si no circula corriente por ninguna de sus bobinas.

En estos motores, las señales de control son trenes de pulsos que van actuando rotativamente sobre una serie de electroimanes dispuestos en el estator. Para conseguir el giro del rotor en un determinado número de grados, las bobinas del estator deben ser excitadas secuencialmente a una frecuencia que determina la velocidad de giro.

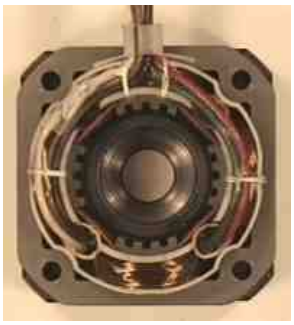

Figura 2.2: Estator de cuatro bobinas

Existen tres tipos de motores paso a paso, también llamados PaP:

- Motores de imanes permanentes.
- Motores de reluctancia variable.
- Motores híbridos.

En los motores de imanes permanentes, el rotor, que posee una polarización magnética constante, gira para orientar sus polos de acuerdo al campo magnético creado por las fases del estator. Este tipo de motor es el más común.

Figura 2.3: Rotor de un motor PaP de imán permanente

En los motores de reluctancia variable, el rotor está formado por un material ferromagnético que tiende a orientarse de modo que facilite el camino de las líneas de fuerza del campo magnético generado por las bobinas del estator.

Los motores híbridos combinan el modo de funcionamiento de los dos tipos anteriores.

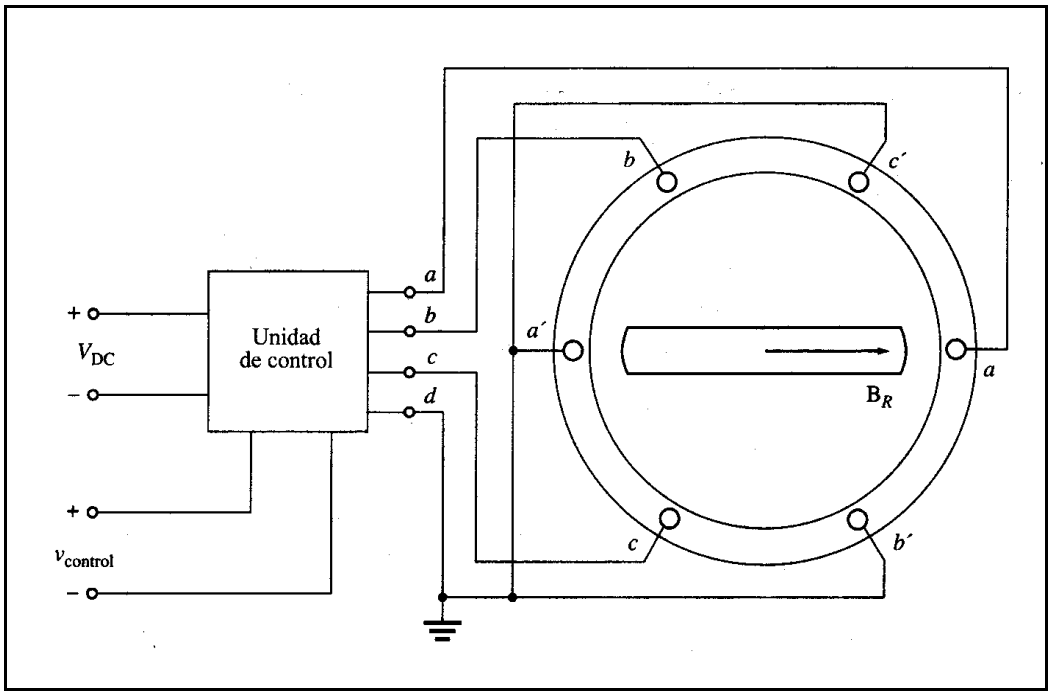

Figura 2.4: Esquema de conexión de un motor PaP de imán permanente

2.3.2 MOTORES SIN ESCOBILLAS

En los últimos 25 años se han ido desarrollando motores que combinan las cualidades de un motor pequeño, como el motor de avance paso a paso de imán permanente, con un sensor de posición del rotor y un circuito de conmutación electrónico.

Los componentes básicos de un motor sin escobillas son:

- Un rotor de imán permanente.
- Un estator con devanado de tres, cuatro o más fases.
- Un sensor de posición del rotor, que suelen ser sensores de efecto Hall.
- Un circuito electrónico para controlar las fases del devanado del rotor.

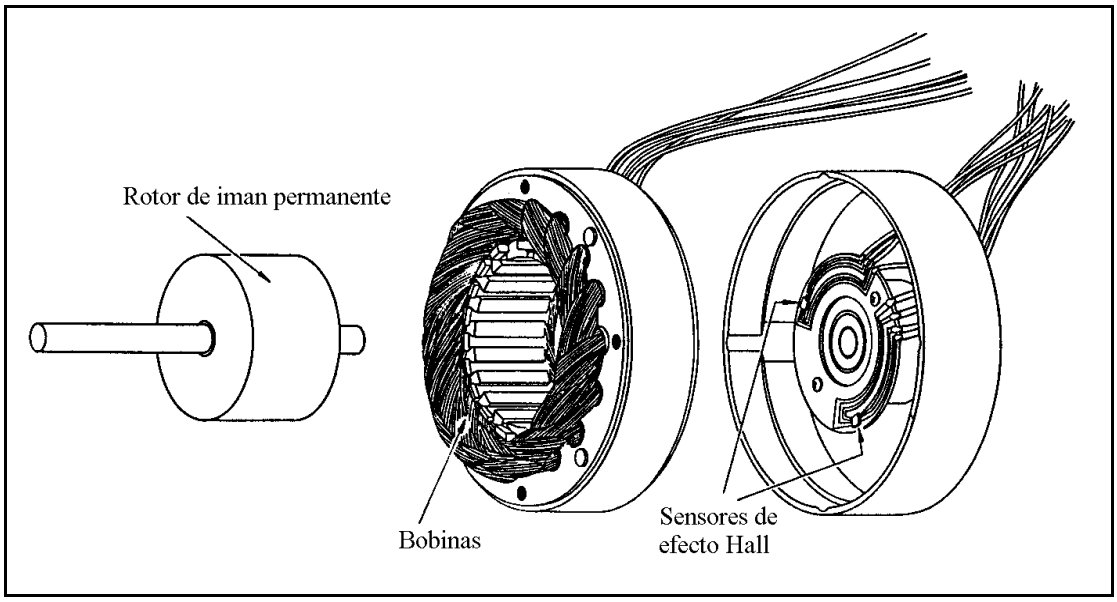

Figura 2.5: Partes de un motor brushless

Éstos motores funcionan aplicando una tensión continua al circuito de control. La electrónica de dicho circuito se utiliza para controlar tanto la velocidad como la dirección del motor. El efecto neto de este diseño, es un motor que opera conectado a una fuente de alimentación de corriente continua, con control total sobre la velocidad y la dirección de la rotación.

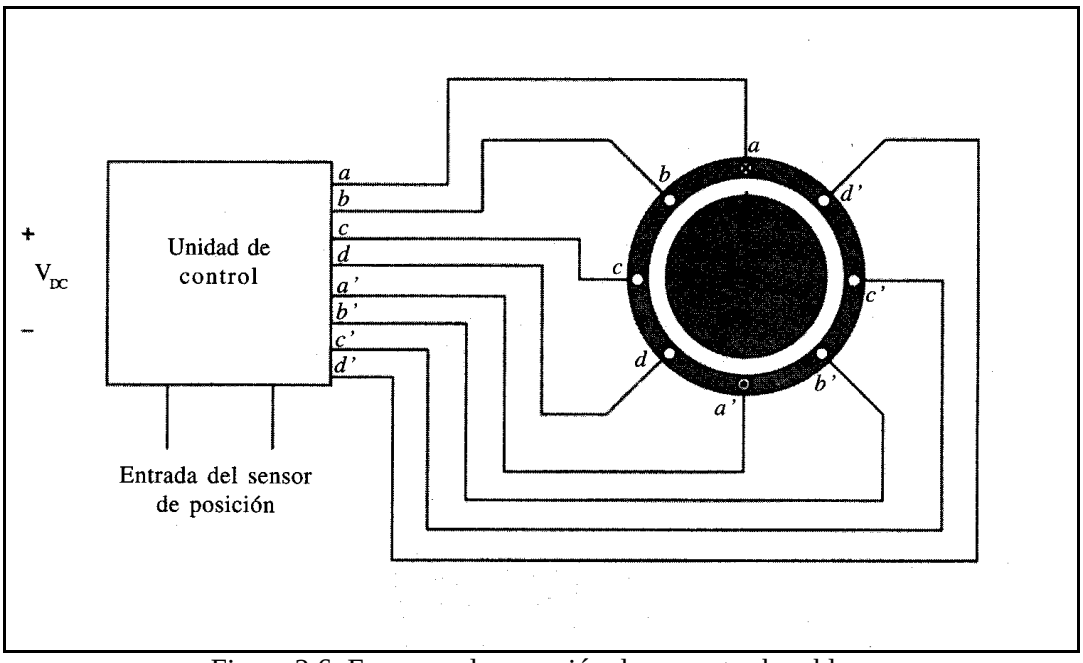

Figura 2.6: Esquema de conexión de un motor brushless

2.3.3 TIPOS DE MOTORES DEL BRAZO DE MANFRED

El robot Manfred utiliza ocho motores sin escobillas; seis motores que mueven las articulaciones del brazo y dos motores que dan movilidad a la base. En este proyecto sólo se trabajará con los motores del brazo.

Los motores utilizados en la implementación del brazo han sido fabricados por la casa Kollmorgen, y pertenecen a la serie RBE(H). Cada articulación del brazo soporta diferentes fuerzas, por tanto, cada una de ellas utiliza un tipo de motor. En la siguiente tabla se especifica el motor y su posición dentro el brazo.

Número de motor	Tipo de motor	Articulación
Motor 1	RBE(H) 01812-B	Hombro (levantamiento frontal del brazo)
Motor 2	RBE(H) 01812-B	Hombro (levantamiento lateral del brazo)
Motor 3	RBE(H) 01214-B	Codo
Motor 4	RBE(H) 00713-B	Giro del antebrazo
Motor 5	RBE(H) 00713-B	Muñeca (arriba, abajo)
Motor 6	RBE(H) 00712-B	Giro de muñeca

Tabla 2.1: Tipos de motores

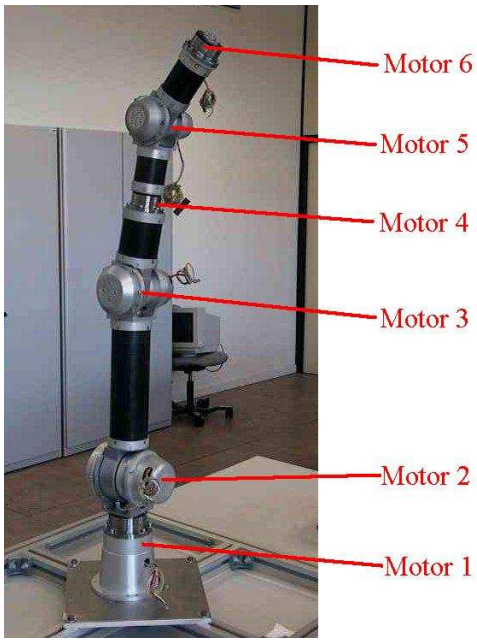

Figura 2.7: Brazo del robot Manfred

2.3.4 PARÁMETROS DE LOS MOTORES DEL BRAZO

Los parámetros de los motores del brazo de Manfred son proporcionados por el fabricante. Éstos varían con el tamaño y modelo del motor. A continuación se definen los parámetros más importantes para la realización de este proyecto.

Par de mantenimiento a temperatura ambiente de 25ºC (Tc): Es el máximo par que proporciona el motor sin que haya rotación.

Par de pico (Tp): Es el máximo par que proporciona el motor, es decir, el par que se genera al aplicar a las bobinas la corriente de pico. La duración máxima es de 10 segundos.

Constante de motor (Km): Es la relación entre el par de pico y la raíz cuadrada de la potencia consumida por el motor (Pp).

$Km = Tp / Pp^{0.5}$

Km se utiliza normalmente para el cálculo de la potencia consumida por el motor a un determinado par.

$Potencia\ consumida = Par^2 / Km^2$

El valor de las constantes de bobinado depende del tipo de bobinado utilizado en cada motor. Según la nomenclatura del fabricante el bobinado de los motores del brazo es de tipo B, característica a tener en cuenta para la realización de los cálculos.

Corriente en par de mantenimiento (Ic): Es la corriente que necesita el motor para generar el par Tc.

Corriente en par de pico (Ip): Es la corriente que necesita el motor cuando genera el par Tp.

Sensibilidad de par (Kt): Es la relación entre el par generado en el motor y la corriente consumida por las bobinas.

La siguiente tabla muestra el valor de los parámetros correspondientes a cada motor.

Parámetros	**01812-B**	**01214-B**	**00713-B**	**00712-B**
Tc (Nm)	1.22	0.467	0.195	0.152
Tp (Nm)	4.62	1.99	0.597	0.447
Km ($Nm/W^{0.5}$)	0.187	0.098	0.047	0.038
Ic (Amp)	2.64	3.73	3.51	3.65
Ip (Amp)	10.0	13.4	10.0	10.0
Kt (Nm/Amp)	0.477	0.132	0.0611	0.0458
Peso (Kg)	1.3	0.641	0.344	0.304

Tabla 2.2. Parámetros de los motores

2.4 REDUCTORES

Dados los parámetros del motor (tabla 2.2), el par máximo obtenido es escaso para las exigencias del brazo. Como resultado, es necesaria la utilización de reductores, que proporcionan una disminución de la velocidad y un aumento del par, dando como resultado un movimiento suave y continuo del brazo.

Los reductores se sitúan a la salida de los motores. Son del tipo harmonic drive, de la casa Harmonic Drive AG.

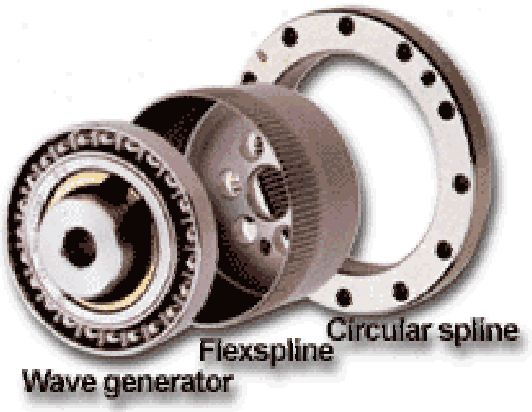

Figura 2.8: Elementos de un harmonic drive

Los elementos que componen un harmonic drive son:

Wave generator: Formado por unas pequeñas bolas, que funcionan como rodamientos. La parte interior del rodamiento de bolas esta fijada al eje, mientras que la parte exterior se deforma a través del rodamiento de bolas. El wave generator se usa normalmente como entrada.

Flexipline: Es una fina pieza de metal en forma de taza con dientes en su cara exterior. Coge la forma del wave generator cuando se unen. La parte de atrás del flexipline es comúnmente denomina diafragma. El diafragma se suele usar como salida.

Circular spline: Es un anillo circular de acero rígido, tiene los dientes en su circunferencia interior. El circular spline normalmente está fijado a la carcasa de la articulación.

2.4.1 TIPOS DE HARMONIC DRIVES DEL BRAZO DE MANFRED

Los harmonic drives que se usan para las articulaciones del brazo son de la casa Harmonic Drives AG, de la serie HFUC-2UH.

Figura 2.9: Harmonic drive (HFUC-2UH)

Al igual que con los motores, cada articulación tiene un tipo diferente de harmonic drive. La figura 2.7 ilustra la localización de cada uno de ellos.

En la siguiente tabla se muestran los tipos de harmonic drives que lleva cada motor.

Número de motor	Harmonic drive
Motor 1	HFUC-25-160-2UH
Motor 2	HFUC-25-160-2UH
Motor 3	HFUC-20-160-2UH
Motor 4	HFUC-17-120-2UH
Motor 5	HFUC-17-120-2UH
Motor 6	HFUC-17-100-2UH

Tabla 2.3: Tipos de harmonic drives del brazo.

Para el correcto desarrollo del proyecto es necesario conocer el valor de algunas de las características de los harmonic drives. La siguiente tabla muestra los valores más relevantes.

Tipo	Relación entre velocidades	Par medido a 2000 rpm (Nm)	Peso (Kg)	Tamaño	Rendimiento a 20ºC
HFUC-25-160-2UH	160	67	1.5	25	78%
HFUC-20-160-2UH	160	40	0.98	20	78%
HFUC-17-120-2UH	120	24	0.64	17	80%
HFUC-17-100-2UH	100	24	0.64	17	81%

Tabla 2.4: Parámetros de los harmonic drives

El fabricante nos proporciona en las hojas de características unas gráficas, de donde se obtiene el rendimiento del harmonic drive. Este dependerá del tipo de lubricación que se use, de la temperatura ambiente y de las revoluciones de la entrada. Según el fabricante, el rendimiento también depende de la carga; esto habrá que tenerlo en cuenta a la hora de calcular el rendimiento real de los harmonic drives. (Anexo B).

2.5 DRIVERS O CONTROLADORES DE MOTORES

Los motores sin escobillas necesitan un circuito electrónico (driver) para poder ser controlados. En el caso de Manfred, cada motor está controlado por un driver comercial.

Los drivers son muy usados en sistemas de control de movimiento, donde es necesario un control de velocidad y posición precisos. El driver transforma las señales de baja potencia, mandadas por la tarjeta controladora, en señales de alta potencia aplicadas a los motores.

Existen muchas formas de amplificar las señales, pero la más eficaz es la modulación de anchura de pulso (PWM). La base del funcionamiento de este sistema es un circuito de control de corriente, el control se consigue variando el ancho de pulso de la señal de salida. Este tipo de amplificadores se usa para motores de corriente continua convencionales.

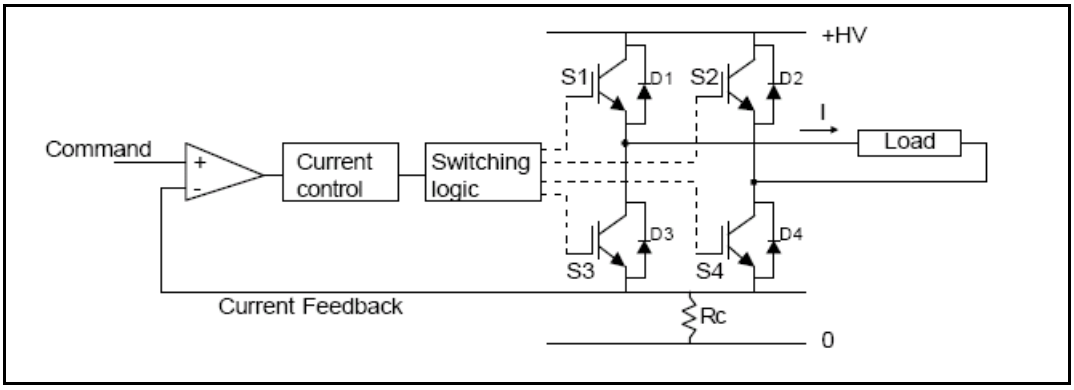

Figura 2.10: Circuito de control PWM para una fase

Para controlar motores sin escobillas existe un amplificador llamado amplificador brushless. Éstos motores requieren una realimentación de posición para su correcto funcionamiento, que consiste en la detección de la orientación del campo magnético del rotor a través de los sensores de efecto Hall. Los sensores están montados de tal forma que generan señales cuadradas con un desfase de 120º.

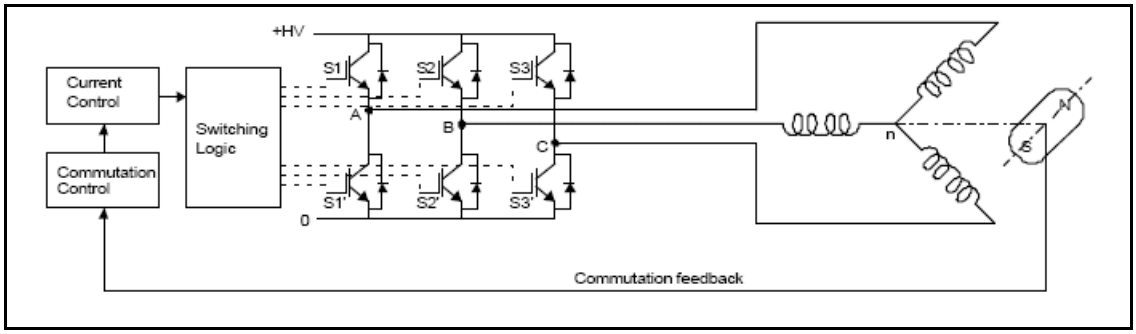

Figura 2.11: Circuito de control para motores brushless

Además de estos tipos de amplificadores, también existen los amplificadores brushless para motores de corriente alterna, que utilizan un encoder o un resolver para la realimentación de posición.

2.5.1 DRIVERS USADOS EN MANFRED

Los drivers que controlan los motores del brazo de Manfred son comerciales, de la casa Advanced Motion Control (AMC), el modelo utilizado es el BE25A20.

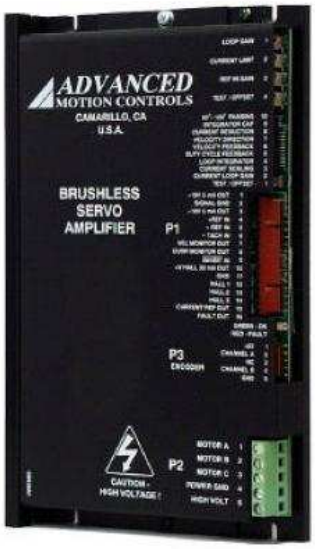

Figura 2.12: Driver ACM - BE25A20

El fabricante indica que este driver puede funcionar en varios modos simplemente ajustando unos interruptores que tiene incorporados. Estos modos son: current mode, encoder velocity mode, open loop mode y tachometer mode.

El driver de cada uno de los motores del brazo de Manfred está ajustado para funcionar en modo “open loop”; esto es, la tensión de salida al motor será proporcional a la tensión de la señal de entrada comandada. Este efecto se obtiene cambiando el ciclo activo de la etapa de salida. El modo “open loop” no tiene realimentación, pues ésta es interna del driver.

Para la realización de este proyecto, el driver dispone de un pin, el P1-8, denominado “current monitor out”, a través del cual, obtenemos una tensión que es proporcional a la corriente que circula por las bobinas, siendo posible de esta manera calcular el par que generan lo motores. La resolución del pin puede ser modificada a través del interruptor 3, llamado “SW3”. Esto permite una medición de la corriente ajustada a las necesidades de tensión especificas del USB-DUX. Según el fabricante, cuando el interruptor se encuentra en posición de apagado se aumenta la resolución, dando una proporción de 1V = 2A; si por el contrario el interruptor está en posición de encendido se disminuye la resolución, dando una proporción de 1V = 4A.

El driver se usa únicamente como amplificador, utilizando los sensores de efecto Hall que incorpora el motor sólo para la excitación secuencial de las bobinas.

Por último, mencionar que Manfred dispone de un encoder acoplado al eje de cada motor para el control de posición y velocidad de los mismos. Este proporciona la cantidad de movimiento realizado a una tarjeta denominada PMAC, encargada de realizar los algoritmos de control para los motores en función de la velocidad y posición de estos.

CAPÍTULO 3
SISTEMA DE ADQUISICIÓN DE DATOS

3.1 INTRODUCCIÓN

Los sistemas de adquisición de datos se utilizan para medir y registrar las señales producidas por transductores. Atendiendo al tipo de señales que se manejan, los sistemas de instrumentación se clasifican en dos tipos: sistemas analógicos y sistemas digitales. Los sistemas analógicos manejan una información esencialmente continua, como por ejemplo una tensión en función del tiempo, etc. Los sistemas digitales utilizan una información representada por una serie de impulsos discretos y discontinuos, cuya evolución en el tiempo expresa la información de la magnitud física que se mide.

Hoy en día, los sistemas de adquisición de datos emplean generalmente señales digitales, ya que éstas ofrecen la ventaja de poder ser procesadas a través del ordenador. El elemento esencial para su funcionamiento es el convertidor analógico-digital (ADC), que mediante una serie de muestreos, convierte la señal analógica en un código digital equivalente.

Del mismo modo, existen los convertidores digital-analógico (DAC), que aceptan una señal digital; código digital, y lo transforman en una salida analógica, que puede ser una tensión o una corriente proporcional al valor digital dado.

3.2 DISPOSITIVO DE ADQUISICIÓN

Para la realización de este proyecto se ha utilizado un sistema de adquisición de datos comercial denominado USB-DUX. De esta manera se cubren las necesidades de resolución y linealidad requeridas por el proyecto; además de los requisitos de trabajar bajo sistema operativo Linux y realizar la comunicación con el ordenador a través de un puerto USB.

Figura 3.1: Interior del USB-DUX

3.2.1 DESCRIPCIÓN GENERAL

El dispositivo USB-DUX cuenta con la flexibilidad y velocidad del puerto USB, pudiendo funcionar tanto con la versión 1.1 como con la versión 2.0 de USB. Los controladores del hardware se encuentran en código abierto para el sistema operativo Linux, siendo compatible con las versiones de la Kernel 2.4.X y 2.6.X.

El muestreo puede ser asíncrono, es decir, el USB-DUX puede obtener los datos en un segundo plano, mientras que la aplicación principal está realizando otras tareas. Este tipo de adquisición permite tener funcionando a la vez hasta 16 dispositivos de adquisición, siempre sujetos a los recursos y a la velocidad del procesador.

Este dispositivo se alimenta a través del puerto USB, por lo tanto, no requiere alimentación externa.

Las entradas y salidas analógicas están eléctricamente desacopladas hasta 1000V.

El dispositivo de adquisición es completamente configurable por software.

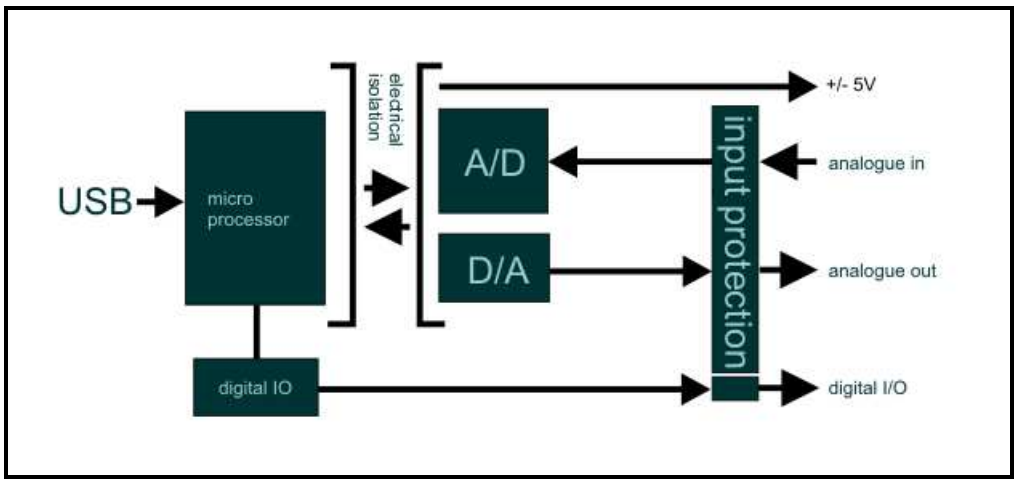

Figura 3.2: Esquemático del dispositivo USB-DUX

El fabricante suministra controladores software y librerías, que ayudan a la instalación y programación del dispositivo, diseñados para el proyecto Comedi, explicado en detalle en el capítulo 4.

3.2.2 ESPECIFICACIONES TÉCNICAS

El USB-DUX cuenta con un circuito conversor analógico-digital, un circuito conversor digital-analógico y un puerto de 8 bits de entrada-salida digital. Las conexiones son del tipo DB-25 y DB-15, es por eso que el fabricante lo suele llamar USB-DUX-D.

Circuito conversor analógico-digital:

8 canales de entrada.

12 bits de resolución por canal.

Rango de entrada (modificable por programa).

- Unipolar: 0...4.096V, 0...2.048V.
- Bipolar: -4.096...+4.096, -2.048...+2.048V.

Resistencia de entrada de 10 Megaohmios.

Frecuencia de muestreo.

- USB 1.1: Hace el muestreo de todos los canales simultáneamente a una frecuencia de 1kHz.
- USB 2.0: Hace el muestreo de cada canal a una frecuencia de 8kHz.

Circuito conversor digital-analógico:

4 canales de salida.

12 bits de resolución por canal.

Rango de salida:

- Unipolar: 0...4.096V
- Bipolar: -4.096...+4.096V.

Los diferentes rangos de salida aparecen en distintos pines del conector.

Corriente máxima de salida de 20 mA.

Escritura asíncrona de todos los canales simultáneamente a 1kHz.

Puerto de 8 bits de entrada-salida digital:

Dirección del canal programable mediante software.

Control de sus dos contadores ascendentes-descendente con “reset”, y frecuencia de muestreo máxima de 500 Hz.

3.2.3 PUERTOS DE CONEXIÓN DEL USB-DUX

El USB-DUX dispone de dos conectores para la entrada y salida de señales. En el conector DB-25 se realiza la conexión de las entradas y salidas analógicas, en este mismo conector se suministra tensiones de +5V y –5V, además de una tensión de referencia de 4,096V. En el conector DB-15 se realiza la conexión de las entradas y salidas digitales, y suministra una tensión de +5V. Para la conexión al ordenador cuenta con un conector USB de tipo B.

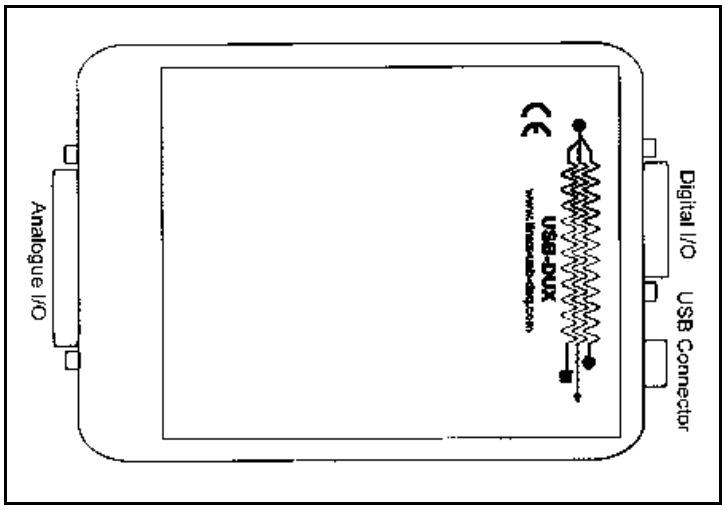

Figura 3.3: Dibujo del USB-DUX

En la tabla 3.1 y 3.2 se muestran los diferentes pines de los conectores así como su función en el USB-DUX.

DIAGRAMA DEL CONECTOR DB-25 (I/O analógica)

Pin	Dirección	Función
1	I	A / D canal 7
2	I	A / D canal 6
3	I	A / D canal 5
4	I	A / D canal 4
5	I	A / D canal 3
6	I	A / D canal 2
7	I	A / D canal 1
8	I	A / D canal 0
9	O	D / A canal 0, bipolar
10	O	D / A canal 1, bipolar
11	O	D / A canal 2, bipolar
12	O	D / A canal 3, bipolar
13	O	V_{ref} = 4.096 V
14	Alimentación	+5V
15	Alimentación	-5V
16	GND	
17	GND	
18	GND	
19	GND	
20	GND	
21	GND	
22	O	D / A canal 0, unipolar
23	O	D / A canal 1, unipolar
24	O	D / A canal 2, unipolar
25	O	D / A canal 3, unipolar

Tabla 3.1

DIAGRAMA DEL CONECTOR DB-15 (I/O Digital)

Pin	Dirección	Función	Contador
1	I/O	Digital I/O bit 7	
2	I/O	Digital I/O bit 6	Reset 1
3	I/O	Digital I/O bit 5	Up / down 1
4	I/O	Digital I/O bit 4	/ CLK1
5	I/O	Digital I/O bit 3	
6	I/O	Digital I/O bit 2	Reset 0
7	I/O	Digital I/O bit 1	Up / down 0
8	I/O	Digital I/O bit 0	/CLK0
9	GND		
10	GND		
11	GND		
12	GND		
13	GND		
14	GND		
15	Alimentación	+5V	

Tabla 3.2

3.3 CONEXIONES UTILIZADAS PARA EL PROYECTO

Para la conexión de los motores al USB-DUX se utiliza únicamente el conector DB-25, pues es el que contiene las entradas del conversor analógico digital. Se ha fabricado un cable, con un conector DB-25 macho, utilizando los pines (8, 7, 6, 5, 4 y 3) para la conexión de las señales provenientes de los drivers de los motores, y los pines (16, 17) como conexiones de masa. Se pueden utilizar más conexiones de masa, pero para este proyecto no es necesario, pues todos los drivers están referenciados a la misma masa.

En la tabla 3.3 se muestra la correspondencia de los cables con los pines usados y con los drivers de los motores a los que van conectados.

Pin DB-25	**Color del cable**	**Número de motor**
3 (A / D canal 5)	Rosa	6 (P1-8)
4 (A / D canal 4)	Violeta	5 (P1-8)
5 (A / D canal 3)	Azul	4 (P1-8)
6 (A / D canal 2)	Verde	3 (P1-8)
7 (A / D canal 1)	Amarillo	2 (P1-8)
8 (A / D canal 0)	Rojo	1 (P1-8)
16 (GND)	Negro	GND (P3-5)
17 (GND)	Gris	GND (P3-5)

Tabla 3.3: Diagrama de conexiones del cable

Como se deduce de la tabla 3.3, el motor 1 está conectado al pin 8 del conector DB-25, que corresponde al canal analógico de entrada denominado 0. Lo mismo ocurre con los demás motores y sus respectivos canales.

Para conectar el cable a los drives de los motores, en el extremo del cable usamos contactos de tipo “molex”.

Como se puede ver en la columna tres de la tabla 3.3, el pin al que va conectado cada cable, es el P1-8 de su driver de motor correspondiente.

Como todos los drivers están referenciados a la misma masa, resulta indiferente usar una conexión u otra. En este proyecto se ha elegido la conexión de masa que se proporciona a través del pin P3-5, pudiendo ser de cualquiera de los drivers.

En la conexión del USB-DUX al ordenador, se usa un cable con un conector USB macho de tipo A, para la conexión al ordenador, y un conector USB macho de tipo B, para la conexión al USB-DUX. Se recomienda que el cable no sea superior a dos metros.

CAPÍTULO 4

COMEDI

4.1 INTRODUCCIÓN

COMEDI son las siglas en ingles de (**co**ntrol and **me**asurement **d**evice **i**nterface). Es un proyecto de software libre, para el control de los dispositivos de adquisición de datos, desarrollado para el sistema operativo Linux. Incluye tres módulos software básicos:

- Un API (**a**plication **p**rogram **i**nterface), que es el conjunto de rutinas protocolos y herramientas para construir programas de aplicación.
- Una colección de módulos para la Kernel de Linux, que implementan este API para dispositivos de diversos fabricantes.
- Una librería de Linux, con funciones y variables especificas de las tarjetas de adquisición, para su programación y configuración.

El proyecto Comedi ofrece controladores software en código abierto para varios dispositivos de adquisición de datos, incluyendo también herramientas y librerías para el manejo de dichos dispositivos. Estos controladores y herramientas vienen distribuidos en dos paquetes software.

Comedi: Es una colección de controladores para múltiples tarjetas de adquisición de datos (que son llamadas dispositivos en la terminología de Comedi). Los controladores se implementan como una combinación de módulos para la Kernel de Linux y unos controladores de bajo nivel. Ambos proporcionan la funcionalidad típica de cada dispositivo.

Comedilib: Es un paquete que se distribuye independientemente del anterior. Contiene librerías que proporcionan una “interface” sencilla a los dispositivos Comedi. Incluidos en este paquete se encuentran la documentación, utilidades de configuración y calibración, así como programas de demostración.

Comedi también es compatible con la aplicación de tiempo real de Linux. Para este propósito se distribuye otro paquete llamado “Kcomedilib”, que contiene los módulos de la Kernel y proporciona el mismo “interface” que Comedilib, pero adaptado para las extensiones de tiempo real de Linux RTAI y RTLinux/Free.

Los controladores suministrados en el paquete “Comedi” son compatibles con las versiones de la Kernel de 2.0.X a 2.6.X.

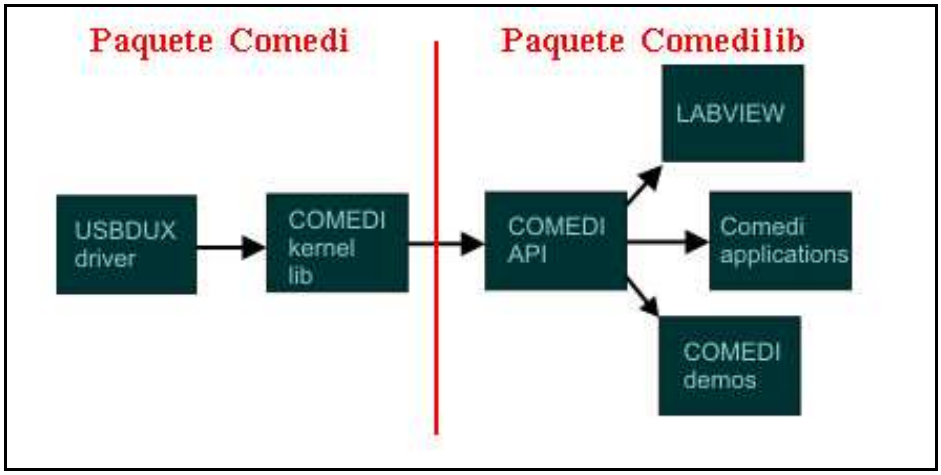

Figura 4.1: Esquema de Comedi

En la figura 4.1 se muestran los tres módulos proporcionados por Comedi para el USB-DUX: USBDUX driver, Comedi kernel lib, Comedi API, y la relación que existe entre ellos. En este proyecto se usa el API de Comedi para programar la aplicación de captura de datos.

Destacar que cualquier programa que utilice las librerías de Comedi, como la aplicación gráfica Labview, podrá funcionar con cualquier dispositivo compatible.

En los siguientes apartados se explica como organiza y controla Comedi a los dispositivos de adquisición de datos compatibles.

4.1.1 SEÑALES DE LOS DISPOSITIVOS

Los dispositivos de adquisición de datos soportados por Comedi pueden tener una o más de las siguientes señales: Entrada analógica, salida analógica, entrada digital, salida digital, entrada de contador, salida de contador, pulso de entrada, pulso de salida.

Entrada-salida analógica. Un canal de adquisición analógico puede ser programado para generar o leer tensiones entre un nivel inferior y un nivel superior (por ejemplo entre -5V y +5V). La electrónica de los diferentes dispositivos puede ser

programada para hacer un muestreo de un grupo de canales en un orden preestablecido y almacenar la secuencia de datos en la tarjeta, o en una parte de la memoria del ordenador usando el acceso directo a memoria o rutinas de interrupción.

Entrada-salida digital. La única configuración que necesitan es definir el número de canales, sus direcciones en el "bus de datos" y su sentido, ya sea de entrada o salida.

Señales basadas en pulsos (contadores, temporizadores, "encoders", etc.) Estas señales son señales digitales en las que también se mide el tiempo entre ellas. Comedi sólo tiene un número limitado de controladores para este tipo de señales.

En complemento a estas funciones, Comedi también ofrece acceso a temporizadores simples.

4.1.2 JERARQUÍA DE DISPOSITIVOS

Comedi organiza todo el hardware de acuerdo con la siguiente jerarquía:

Channel: O canal. Es el nivel hardware más bajo. Representa las propiedades de un sólo canal de datos, por ejemplo, una entrada analógica, o una salida digital. Cada canal tiene varios parámetros: Como el rango de tensiones, la tensión de referencia, la polaridad del canal (unipolar o bipolar), el factor de conversión entre tensión y unidad física, los valores binarios "0" o "1", etc.

Subdevice: Define a un grupo de canales idénticos que están físicamente implementados en la misma tarjeta. Por ejemplo, un grupo de 8 salidas analógicas idénticas. Cada subdevice tiene parámetros para el número y el tipo de canal.

Device: O dispositivo. Define a un grupo de subdevices que están físicamente implementados en la misma tarjeta, en otras palabras, la tarjeta de adquisición de datos por ella misma. Por ejemplo, el device USB-DUX tiene un subdevice de ocho canales de entrada analógica, otro subdevice de cuatro canales de salida analógica y un tercer subdevice con ocho líneas de entrada y salida digital.

Algunas tarjetas de adquisición tienen componentes extras que no se ajustan a lo mencionado en la clasificación anterior, como por ejemplo una EEPROM para guardar la configuración y los parámetros de la tarjeta, o señales de calibración. Estos componentes especiales son también clasificados como subdevices en Comedi.

4.1.3 TERMINOLOGIA DE ADQUISCIÓN

Esta sección introduce la terminología que se usa cuando hablamos de adquisición. La figura 4.2 muestra una secuencia típica de adquisición.

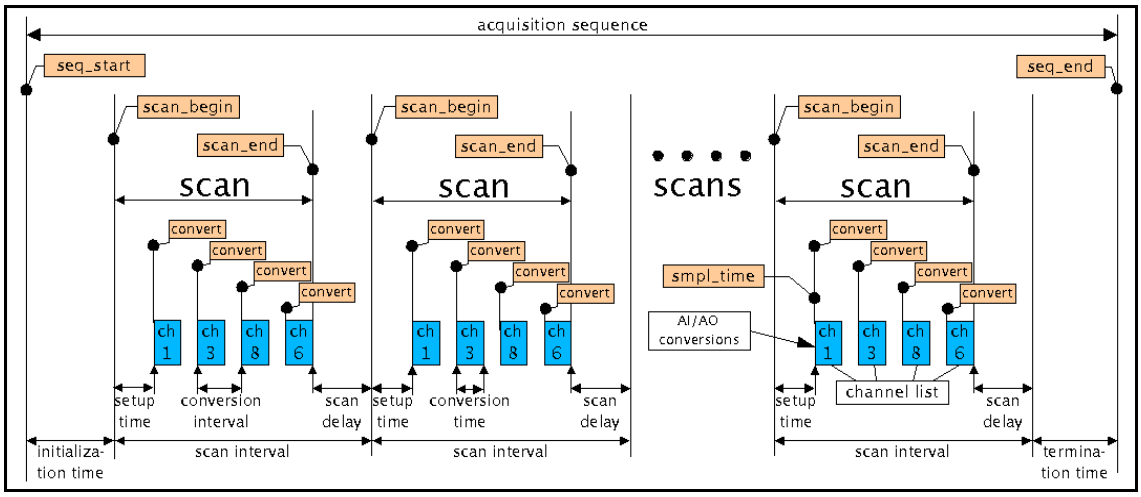

Figura 4.2: Secuencia de adquisición

Toda adquisición de datos tiene un principio y un final, representados en la figura 4.2 como *seq_start* y *seq_end* respectivamente. En cada una de estas partes, tanto el hardware como el software necesitan unos tiempos de inicialización, representados como *initialization time* y *termination time*. La secuencia de adquisición se compone de una serie de muestreos, que se representan en la figura 4.2 como *scans*. En esta parte es donde se hace la adquisición de datos; los datos son leídos de la tarjeta o escritos en ella. Cada muestreo o *scan* tiene, a su vez, un principio y un final, representados en la figura como *scan_begin* y *scan_end* respectivamente. Al tiempo de inicialización del muestreo se le denomina *setup time*. Existe un tiempo de configuración, al final de cada muestreo o *scan*, al que se le denomina *scan delay*.

Se llama *scan intreval* al tiempo necesario para completar un muestreo, es decir, el tiempo que transcurre desde que empieza un muestreo hasta que empieza el siguiente.

En cada muestreo o *scan* se realizan las conversiones de los canales seleccionados. Por ejemplo, el convertidor analógico-digital o digital-analógico es activado para cada uno de los canales programados, y al tiempo que éste tarda en convertir los datos de cada canal se le denomina *conversión time*, empezando desde el instante de tiempo denominado *sample time* (viene escrito como *smpl_time*). Cada conversión es producida por un evento de disparo, ya sea externo o interno, en la figura 4.2 este evento viene descrito como *convert*. Cada hardware tiene diferentes tiempos de conversión, este tiempo es el tiempo mínimo transcurrido entre conversiones. En la figura 4.2 se le denomina *conversión interval*.

Algunas tarjetas de adquisición deben multiplexar las diferentes conversiones sobre un sólo convertidor analógico-digital o digital-analógico, de esta forma, las conversiones se hacen en serie, como la mostrada en la figura 4.2; otras tarjetas tienen dos o más convertidores, esto le permite hacer conversiones en paralelo.

En general, no sólo el principio de la conversión esta forzado, también el principio de cada adquisición o *scan* y el principio de la secuencia o *adquisition sequence*. Comedi permite configurar que fuente se quiere usar como disparo, tanto para la conversión como para la adquisición. También permite especificar la lista de canales a los que se les hace el muestreo.

4.1.4 TIPOS DE FUNCIONES PARA LA ADQUISICIÓN DE DATOS

En esta sección, pasamos a explicar los cuatro modos de funcionamiento que nos ofrece Comedi para la adquisición o la escritura de datos.

Adquisición simple: Funciones para obtener un sólo dato en un solo canal. Estas funciones trabajan de forma síncrona, esto significa que el proceso llamante es bloqueado hasta que la adquisición es terminada. En este tipo de funciones se encuentran las funciones *comedi_data_read(), comedi_data_write(), comedi_dio_read(), comedi_dio_write().*

Instrucciones: Realizan adquisición de datos múltiples o simples en un determinado canal de manera síncrona. Por lo tanto, las llamadas a estas funciones bloquean el proceso hasta que ha finalizado toda la adquisición. Por ejemplo: *comedi_do_insn().*

También se pueden ejecutar listas de instrucciones (en diferentes canales) en una sola llamada. De esta forma se reduce el trabajo de configuración de cada adquisición individual, por ejemplo: *comedi_do_insnlist().*

Comandos: Un comando es una secuencia de *scans* para la cual se han especificado condiciones que determinan cuando empieza y termina la adquisición de datos.

La llamada a la función *comedí_command()* genera adquisición de datos asíncrona. Tan pronto como la información del comando ha sido rellenada, la llamada a la función *comedi_command()* regresa. El hardware de la tarjeta se encarga de la secuencia y tiempos de adquisición, y se asegura que los datos adquiridos son entregados en un buffer software definido por el proceso llamador. Las operaciones asíncronas requieren alguna forma de funcionalidad de "callback" para evitar el "buffer overflow".

Las funciones aquí explicadas, sólo son algunas de las funciones que utiliza Comedi para la adquisición de datos. (Para más referencia ver Anexo E).

CAPÍTULO 5
PROGRAMA PARA LA ADQUISICIÓN DE LOS DATOS CON USB-DUX

Para la obtención de los datos a través del dispositivo USB-DUX, ha sido necesario la creación de un programa en lenguaje C. En la realización de este, se han usado librerías estándar de C, exceptuando la librería llamada “comedilib.h”, proporcionada en la instalación de Comedi, siendo esta necesaria para las funciones que controlan la tarjeta de adquisición de datos.

5.1 DESCRIPCIÓN DEL PROGRAMA

El programa creado para la obtención de los datos de los seis motores del manipulador móvil, realiza un muestreo, durante un tiempo indeterminado, de los canales seleccionados del USB-DUX, calcula el par que generan las articulaciones y una vez que el usuario decide parar la adquisición pulsando la tecla “intro”, guarda los datos en un archivo llamado “motores” y muestra una representación gráfica de los valores obtenidos, mediante la ejecución del comando “gnuplot”.

Cuando el programa es ejecutado se crean dos procesos diferentes, uno es el encargado de adquirir y mostrar en pantalla los datos, y el otro, simplemente espera a que el usuario pulse “intro” para mandar una señal de terminación al proceso de adquisición y finalizar el programa.

En el anexo C, se muestra el diagrama de flujo, donde se puede observar la estructura de programación utilizada para la realización del programa.

El proceso denominado “padre” es el encargado de la adquisición y presentación, mientras que el proceso denominado hijo, es el que espera la pulsación de la tecla.

5.1.1 PROCESO PADRE

En este proceso se han programado diferentes funciones, tales como: la obtención de los datos, el filtrado, la conversión del dato a un valor numérico de tensión, el cálculo del par generado por cada motor, la visualización del dato por pantalla y la representación gráfica del mismo.

En este apartado solo se explicarán las funciones especiales de Comedi que utilizamos en el programa. Estas son funciones de adquisición y conversión a un valor numérico de tensión.

Apertura del dispositivo: Se realiza mediante una función de Comedi. En el código fuente del programa, se utiliza la siguiente línea.

```
cosa=comedi_open("/dev/comedi0");
```

Esta función abre el dispositivo conectado en "/dev/comedi0", y le asigna el nombre "cosa" para poder acceder a él.

Adquisición de los datos: Esta función necesita unas variables especiales, tales como el dispositivo y el subdevice que se utilizarán, el canal que se va a leer, el rango de tensiones del canal, la referencia de masa usada en las señales analógicas y la variables donde se guardarán los valores.

```
comedi_data_read(cosa,subdev,canal,rango,referen, & dato[canal]);
```

Al principio de la función "padre" se pueden ver las siguientes definiciones de variables.

```
int subdev = 0;
int canal = 0;
int rango = 2;
int referen = AREF_GROUND;
```

La primera variable es utilizada para especificar a las funciones de adquisición, el número de subdevice del USB-DUX que se utilizará, que en este caso es el 0; las entradas analógicas. (Para más información acerca de los subdevices ver anexo E). Con la segunda variable se define el canal que se va a leer; el programa empieza en el canal analógico 0 y termina en el 5. También se concreta el rango de tensiones en el que se trabaja; para este caso, se trabajará de 0 a 4,096 V que corresponde al valor 2 de la variable *rango.* También es muy importante definir la referencia a masa que se va a tomar para la lectura de los datos; en este caso, la referencia está tomada de la señal de masa que se introduzca al USB-DUX a través del puerto de señales analógicas.

En la tabla 5.2 se muestran los posibles valores de las variables *rango* y *referen.*

VALORES DE LA VARIABLE "RANGO"	**RANGO DE TENSIONES**
0	-4,096 a +4,096
1	-2,048 a +2,048
2	0 a +4,096
3	0 a +2,048

Tabla 5.1: Rango de tensiones del USB-DUX

VALORES DE LA VARIABLE "REFEREN"	**USOS**
AREF_GROUND	Para entradas-salidas referenciadas a masa
AREF_COMMON	La referencia es común a todos los canales pero está aislada de masa
AREF_DIFF	Para entradas-salidas diferenciales
AREF_OTHER	Para referencias que no se ajustan a ninguna de las anteriores

Tabla 5.2: Valores de la referencia a masa

Conversión de los datos a un valor numérico con unidades físicas: Para esta función necesitamos otras dos más, una función que nos indique cual es el valor máximo analógico del canal y otra que nos indique todo lo necesario para saber el rango que estamos utilizando.

Para saber el valor máximo que podemos obtener en un canal se ejecuta el comando:

```
maxdato=comedi_get_maxdata(cosa,subdev,canal);
```

Se ha de especificar el dispositivo a usar, el subdevice y el canal de donde se obtendrá el dato. En este caso, todos los canales tienen el mismo rango, por lo tanto, el dato máximo será siempre el mismo; por eso, esta función es ejecutada una sola vez.

Para obtener el rango de tensiones en el que se trabaja, se utiliza el siguiente comando.

```
tiporango=comedi_get_range(cosa,subdev,canal,rango);
```

En esta función se ha de especificar el dispositivo a usar, el subdevice, el canal de donde se obtienen los datos y el rango utilizado para obtenerlos. Esta función, representa la información para la conversión.

Como ocurre con el anterior, también se ejecuta una sola vez.

Una vez ejecutados estas funciones, se puede pasar a la conversión de los datos. Para eso se utiliza el siguiente comando.

```
voltios[canal]=comedi_to_phys(dato[canal],tiporango,maxdato);
```

Esta función devuelve el dato que se pasa como argumento, convertido en una unidad física, con un valor comprendido entre 0 y “maxdato”.

Una vez se termina de convertir el dato, el programa pasa a calcular la corriente equivalente a la tensión obtenida. Como se dijo en capítulos anteriores, el driver de los motores da una tensión proporcional a la corriente que consume el motor, por lo tanto solo habrá que multiplicar el dato por el factor de proporcionalidad; en este caso, la resolución del pin es 1V = 4A, por lo tanto, la constante de proporcionalidad es 4. Esta operación se ha programado mediante una macro, puesto que su ejecución es mucho más rápida que la de una función.

A la hora de calcular el par, cada motor tiene diferente sensibilidad de par y cada harmonic drive diferente rendimiento y relación de par, por lo tanto, el programa distingue si el dato pertenece al motor 1 o al motor 6, para mediante otra macro, calcular el par generado en cada articulación.

Presentación en pantalla: Los datos obtenidos en las anteriores operaciones son presentados por pantalla y guardados en un archivo llamado “motores”.

Mientras dura la adquisición, se presentan en pantalla los datos ordenados por columnas, perteneciendo la primera columna al primer motor, la segunda al segundo motor y así sucesivamente.

Cuando el usuario determine que debe parar la adquisición, basta con pulsar la tecla intro, mostrándose en la pantalla una representación gráfica de los datos obtenidos.

Por último, habrá que tener especial cuidado cuando se quieran comparar dos movimientos diferentes, pues el programa rescribirá el archivo “motores” con los datos de cada nueva adquisición. Si se desea comparar dos movimientos, simplemente habrá renombrar el archivo “motores” creado, antes de ejecutar de nuevo el programa.

5.1.2 PROCESO HIJO

Este proceso espera a que se pulse la tecla (intro) o llegue un final de línea, que se puede generar pulsando (control + d). Una vez se pulsa la tecla (intro) o (control + d), el proceso hijo sale del estado de espera, y manda al proceso padre una señal, indicándole que debe terminar el muestreo.

5.1.3 REPRESENTACIÓN GRÁFICA

Para la representación gráfica se usa el comando “gnuplot”. Este comando representa mediante una gráfica los valores que le pasemos como argumento. Para la representación de los datos de los motores se ha creado un archivo llamado “graficas”. En el se especifica como se han de representar los datos obtenidos, que han sido guardados en el archivo “motores”. Será necesario tener el archivo “gráficas” en el mismo directorio donde se encuentre el ejecutable del programa de adquisición.

Las líneas contenidas en él son las siguientes:

```
plot [ ][ ] 'motores' u 1 title 'MOTOR 1 (Nm)' with lines, 'motores' u 2 title 'MOTOR 2 (Nm)' with lines, 'motores' u 3 title 'MOTOR 3 (Nm)' with lines, 'motores' u 4 title 'MOTOR 4 (Nm)' with lines, 'motores' u 5 title 'MOTOR 5 (Nm)' with lines, 'motores' u 6 title 'MOTOR 6 (Nm)' with lines
```

Al terminar la adquisición, se crea una ventana donde se ve la representación gráfica de los valores. Cada gráfica corresponde a un motor; el motor al que pertenece, aparece indicado en la leyenda.

5.2 COMPILACIÓN DEL PROGRAMA

Para compilar el programa se usa un compilador de C. Puesto que las librerías de Comedi no son estándar, ha de ponerse una opción llamada “-lcomedi”, en la compilación, que especifica al compilador que use esas librerías en el proceso de enlazado.

CAPÍTULO 6
PRUEBAS REALIZADAS Y SIMULACIÓN MEDIANTE MATLAB

6.1 INTRODUCCIÓN

Previamente a las pruebas realizadas con Manfred se comprobó experimentalmente la respuesta del USB-DUX con una fuente de tensión continua. Para ello, se midió el valor de tensión entregado por la fuente de alimentación con un voltímetro y se observo la correspondencia con el valor obtenido por el USB-DUX. Para el voltímetro, la máxima resolución es de 1 mV, el USB-DUX tiene 12 bits de resolución, con una tensión máxima de 4,096 V, por lo tanto, la resolución máxima también será de 1 mV. El resultado de esta prueba fue satisfactorio, obteniendo idéntico resultado en ambos.

En las pruebas de laboratorio se ha utilizado exclusivamente el motor de la articulación 3, pues es el más sencillo en movilidad y en cálculo de pares creados por los pesos del brazo. También se ha realizado una prueba de movimiento usando la rutina de "home", esta rutina mueve las articulaciones una a una hasta situarlas en su posición inicial.

Para la comprobación de los datos se ha calculado el par creado por el peso del brazo (anexo A).

6.2 PRUEBAS DE MOVIMIENTO

Las pruebas consisten en la digitalización de la tensión que se obtiene en el pin "P1-8" del driver del motor correspondiente mediante el USB-DUX, para luego calcular el par instantáneo generado por la articulación, tanto si está en movimiento como si está enclavada.

Cuando el brazo está enclavado, los motores tienen que soportar el par que genera el peso de las articulaciones; así pues, el motor evita la caída de la articulación, debido a la gravedad. Si la articulación está es movimiento, el par dependerá de la velocidad que se le imprima.

Como se verá más adelante, la mayoría de las pruebas están realizadas desde la posición inicial (0º desde la vertical), a la posición final (90º desde la vertical), exceptuando la prueba de sensibilidad y la prueba generada con la rutina de movimiento "home".

Cabe destacar que debido a la no-linealidad en el rendimiento de los harmonic drives, es complicado saber exactamente su rendimiento, y por lo tanto, el par de la articulación, en este caso el rendimiento utilizado es el rendimiento que se ha calculado en el Anexo B.

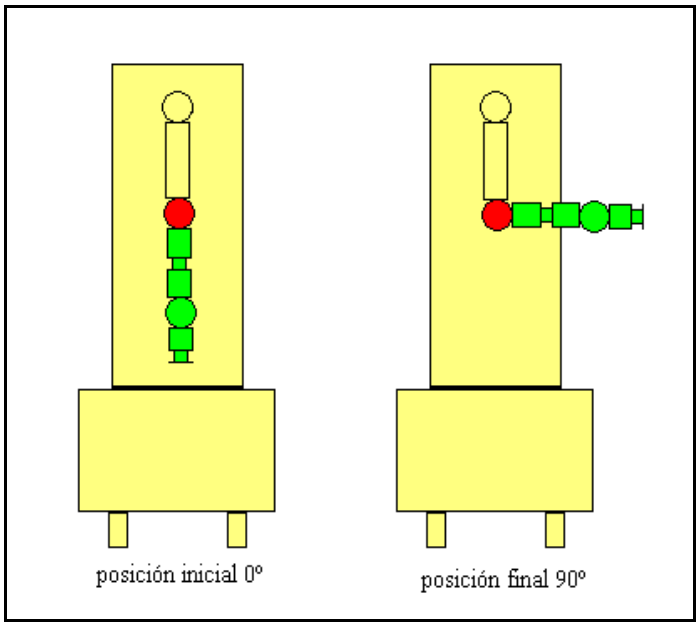

Figura 6.1: Posiciones inicial y final

Otro aspecto a tener en cuenta es la necesidad de enclavar la articulación 1, para que cuando se produzca el movimiento de la articulación 3, no cometer errores de interpretación de los datos obtenidos.

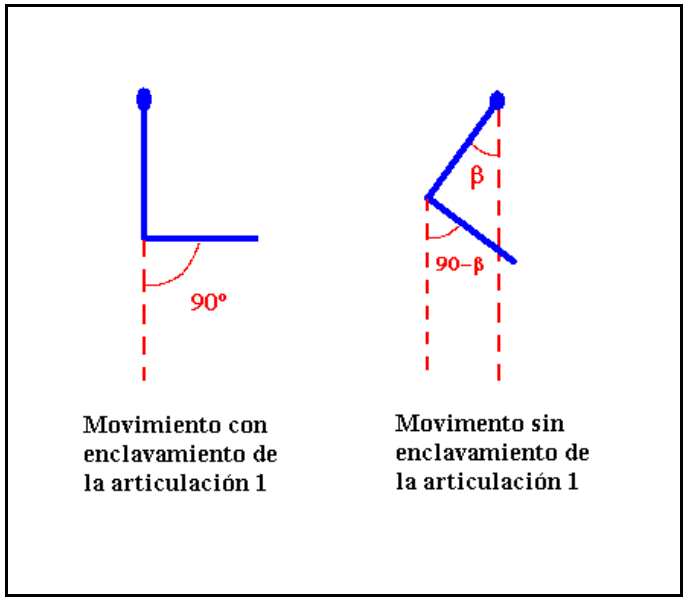

Figura 6.2: Diferencia de posición de las articulaciones

En la figura 6.2 se puede observar un esquema de la variación del ángulo de la articulación 3 con respecto a la vertical, provocado por la variación del centro de gravedad del brazo.

6.2.1 MOVIMIENTO DE ASCENSO Y DE DESCENSO.

Antes de pasar a la gráfica con el par generado por la articulación, decir, que la prueba consistió en un primer movimiento, desde la posición inicial a la posición final, para después mantener parada la articulación durante un tiempo en la posición final, y por último mover el brazo de la posición final a la inicial.

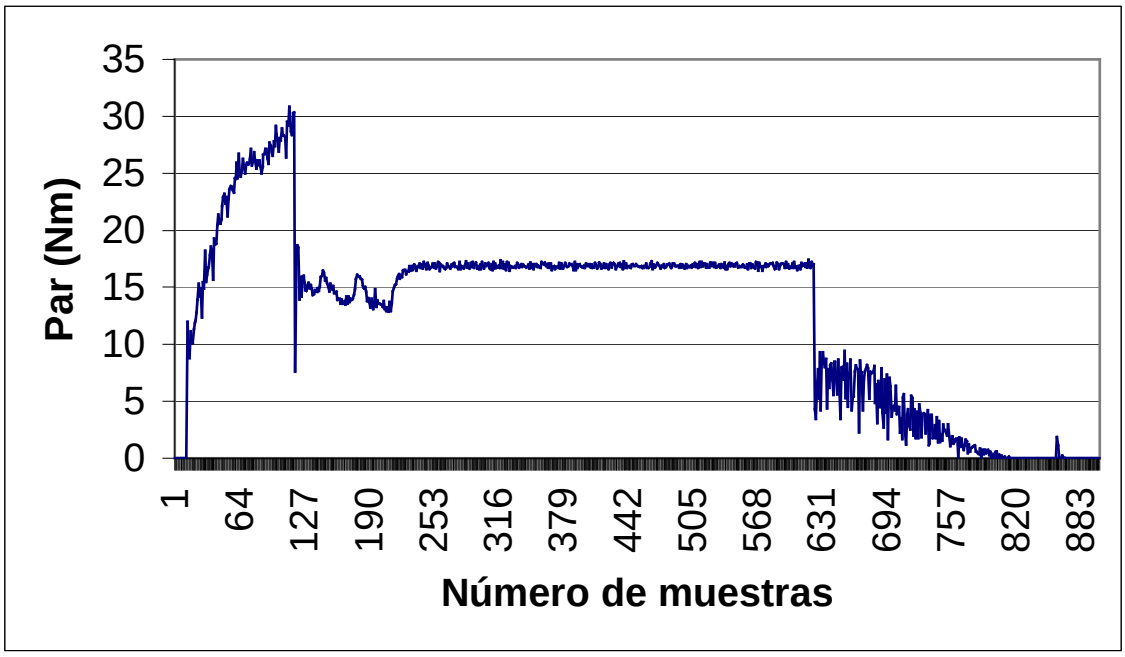

Figura 6.3: Par de ascenso y descenso

En esta prueba, se observa un incremento instantáneo del par de 0 a 10 Nm, este par es generado para vencer el rozamiento estático de la carga. Una vez ha empezado, se observa que el motor aumenta de par a medida que aumenta la posición de la articulación, este aumento se produce a través de la compensación que hace la tarjeta que controla los motores (PMAC) para mantener la velocidad de la articulación constante. En este tramo de gráfica se generan hasta 30 Nm cuando la articulación llega a la posición final.

Si se observa el tramo de gráfica donde el motor estuvo enclavado, en primer lugar llama la atención unas pequeñas variaciones al principio, estas variaciones se deben al movimiento de la articulación 1 en la posición final. Si la articulación 1 no compensa la fuerza generada en ella, el ángulo de la posición final de la articulación 3 disminuirá, y por lo tanto la lectura ya no será la correspondiente a la posición final (figura 6.2). Las fluctuaciones en este pequeño tramo se produjeron hasta que el motor 1 recibió la orden de compensar la fuerza ejercida en él. A continuación de las fluctuaciones, se puede ver un tramo donde el par es constante, este corresponde a la posición final, ejerciendo la articulación un par de 16 Nm, que como se comprueba en el anexo A, es el par máximo que tiene que soportar dicha articulación, debido al peso del brazo.

El último tramo es un movimiento descendente, de la posición final a la inicial. En este tramo el motor sólo tendrá que frenar al brazo, para que caiga con la velocidad deseada; se observa que el motor libera y sujeta al brazo constantemente.

6.2.2 MOVIMENTO CON PARADA INTERMEDIA

En esta prueba no se sujeto la articulación 1 para que compensase la fuerza debida al cambio del centro de gravedad.

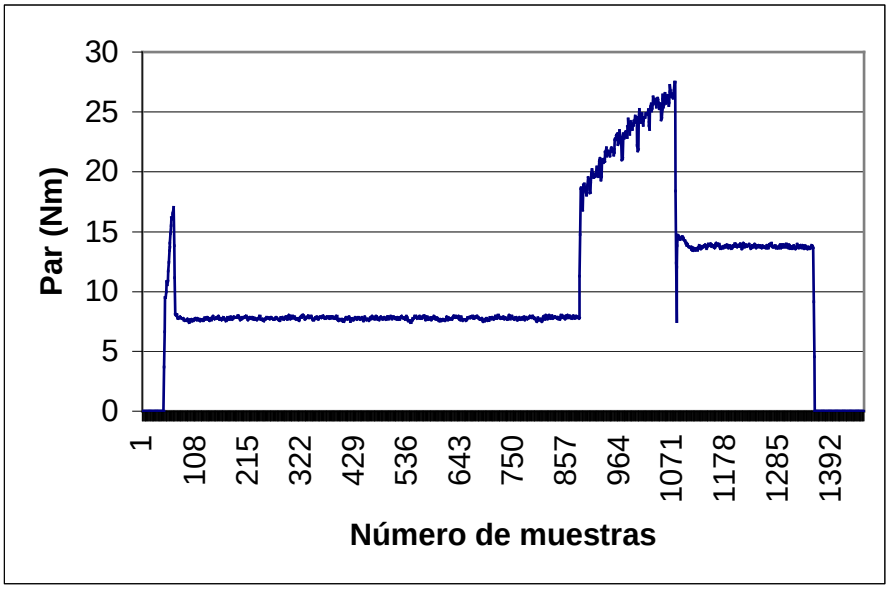

Figura 6.4: Par en movimiento con parada intermedia

Como se observa en la gráfica, hay un primer movimiento en el que el motor empieza a moverse hasta que llega aproximadamente a los 45º, es aquí, donde el par

generado por el motor baja de 16 Nm cuando se está moviendo, a aproximadamente 8 Nm cuando el motor se detiene, esta disminución está provocada por que ya no existe coeficiente de rozamiento dinámico en la carga. Teóricamente, el par que tendría que soportar la articulación a 45º sería de 11 Nm, esta diferencia se debe a la variación del ángulo de la articulación 1 con respecto a la vertical (figura 6.2). Con los 8 Nm obtenidos, mediante cálculos teóricos, se determina que el ángulo para el cual la articulación generaría dicho par es de 30º, por lo tanto, sabemos que la articulación 1 habrá variado unos 15º con respecto a la vertical.

En el siguiente tramo de la gráfica, el motor empieza a moverse otra vez, ahora desde los 45º hasta los 90º, nótese que el motor sube instantáneamente de 8 Nm hasta aproximadamente 17 Nm, este incremento instantáneo es producido por la necesidad de vencer el par estático de la carga.

En el segundo movimiento, el par sube hasta los 26 Nm al alcanzar el ángulo de 90º. Cuando el motor llega a la posición horizontal, se detiene. En esta posición la gráfica muestra que el par cae hasta los 14 Nm. Ocurre lo mismo que en la anterior posición estática, el par generado es menor que el calculado teóricamente; en este caso se obtiene que el ángulo que habrá variado la articulación 1 con respecto a la vertical, es de unos 30º.

6.2.3 MOVIMIENTO CON PERTURBACIONES

Se realizó para ver la respuesta de la articulación en movimiento, aplicando pares puntuales tanto a favor como en contra del movimiento.

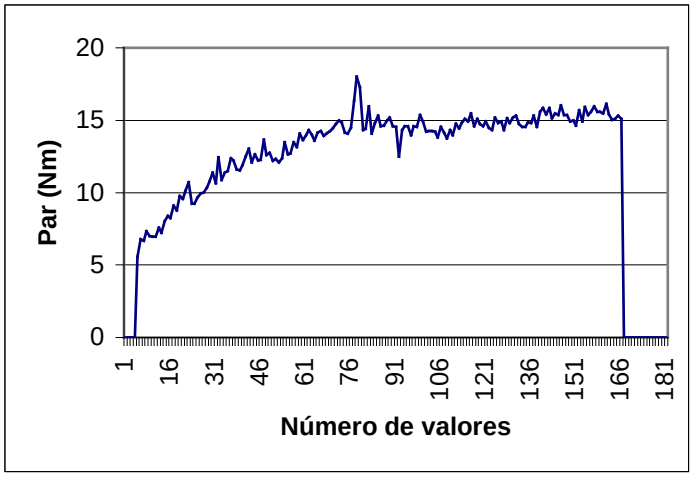

Figura 6.5: Reacción del motor 3 a las perturbaciones

En esta prueba, la articulación se mueve desde los 0º hasta los 85º, el motor no llegó a los 90º pues su velocidad era demasiado lenta.

En la gráfica, se pueden ver dos picos diferenciados, primero se observa un pico que sube hasta los 18 Nm, en este punto es cuando se le aplicó el par resistente, este consistió en un pequeño golpe opuesto al movimiento. Seguidamente en la gráfica se observa un pico que baja hasta 12,5 Nm, esto es debido a un pequeño empujón a favor del movimiento, que se dio a la articulación.

De esta prueba se concluye que el motor en movimiento podrá responder con eficacia a las fuerzas externas puntuales que se le apliquen, pero no podrá responder a las fuerzas externas aplicadas de forma continua si la velocidad de movimiento es muy baja para soportar dichos pares. Esto viene influenciado por los límites que tiene programado el controlador de los motores para no dañarlos.

6.2.4 REACCIÓN AL AUMENTO EXTERNO DE PAR EN ESTÁTICA

En esta prueba, manteniendo enclavado el motor 1, se movió la articulación 3 hasta que formó un ángulo de 90º con la vertical y se enclavó en esa posición, a continuación se colgaron pesos para ver su respuesta.

Se realizaron dos pruebas diferentes, en la primera se tomaron las muestras a la frecuencia estándar del USB-DUX y la segunda se tomó una muestra cada 5 segundos.

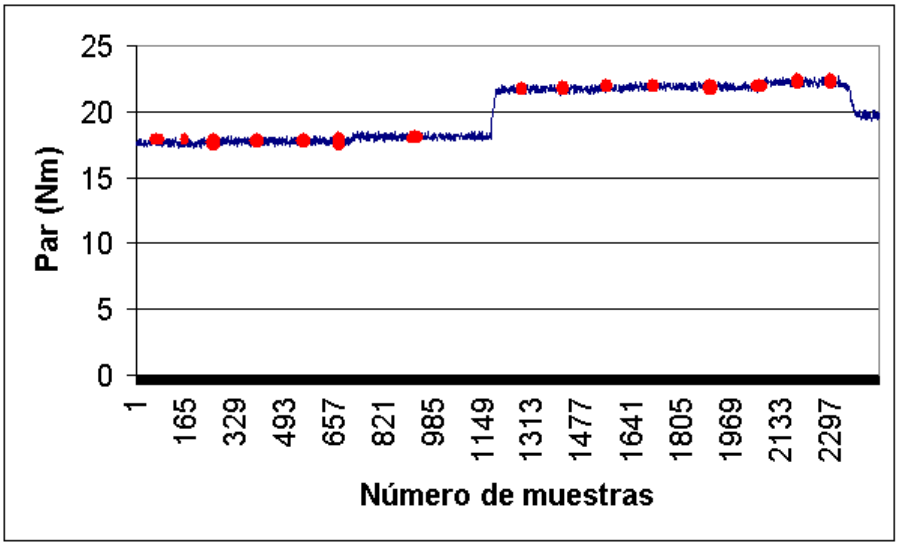

Figura 6.6: Respuesta del motor en estática (prueba 1)

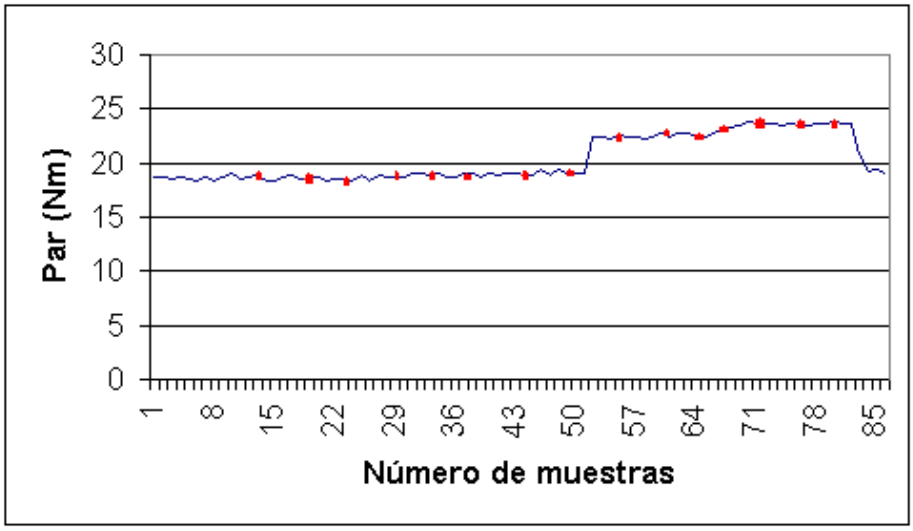

Figura 6.7: Respuesta del motor en estática (prueba 2)

En la primera prueba cabe destacar que no se empieza desde los 16 Nm que debería ejercer el motor al peso de las articulaciones, sino que se empieza a partir de los 17 Nm. Esta pequeña diferencia se debe a la propia configuración del brazo, pues el par que se calcula teóricamente es una aproximación, y a la hora de obtener los pares experimentales, habrá que tener en cuenta el error cometido en la lectura.

En la segunda prueba, se empezó desde los 18 Nm, esto se explica observando la grafica de la primera prueba; si observamos el final de la gráfica, cuando se quitó todo el peso externo aplicado, el par no disminuyó hasta los 17 Nm del principio, sino que se quedó en 20 Nm, esto se debe a que la tarjeta responde mejor a un incremento de par externo que a una disminución. Se comprobó que después de pasado un tiempo de haber dejado al brazo sin pesos, el par volvía al valor inicial 17 Nm, pero este tiempo ronda los 4 minutos.

En la gráfica se han marcado unos puntos rojos. Estos puntos rojos marcan el momento donde se colgó el peso adicional; nótese que hay 15 puntos y cada punto aumenta el peso del brazo en 100 gramos, hasta un máximo de 1500 gramos. Los pesos fueron colgados a una distancia de 48 cm del motor 3, obteniendo los siguientes resultados.

Peso añadido (gr)	**Par teórico del peso añadido (Nm)**	**Par teórico total que sujeta el motor (Nm)**	**Par experimental (prueba 1) que sujeta el motor (Nm)**	**Par experimental (prueba 2) que sujeta el motor (Nm)**
100	0,4704	16,4704	17,557056	18,741888
200	0,9408	16,9408	17,557056	19,065024
300	1,4112	17,4112	17,664768	19,226592
400	1,8816	17,8816	17,718624	19,11888
500	2,352	18,352	17,718624	19,442016
600	2,8224	18,8224	17,5032	19,442016
700	3,2928	19,2928	18,257184	19,334304
800	3,7632	19,7632	21,219264	19,711296
900	4,2336	20,2336	21,5424	22,8888
1000	4,704	20,704	21,973248	23,15808
1100	5,1744	21,1744	21,703968	23,15808
1200	5,6448	21,6448	21,919392	23,588928
1300	6,1152	22,1152	22,134816	24,073632
1400	6,5856	22,5856	22,296384	24,019776
1500	7,056	23,056	22,565664	24,342912

Tabla 6.1: Respuesta del motor en estática

En los datos de la primera prueba, se observa que el par permanece constante a 17 Nm hasta que se le aplican 700 gramos, lo que incrementa el par en 0,75 Nm, y que se genera un salto en el par de unos 2 Nm a partir de aplicarle un peso adicional de 800 gramos. Esto sugiere que fue en ese instante, cuando el peso total del brazo venció el par resistente que generaba la articulación, y por consiguiente, tuvo que aumentar el par para que no descendiera el brazo.

En la segunda prueba existe también un salto de 2 Nm cuando le aplicamos 900 gramos. Ahora el motor tarda más en reaccionar, pues a pesar de tener las mismas condiciones de la prueba anterior, está generando más par, debido a que se realizó a continuación de la primera, sin mover el brazo.

De los datos en la tabla se deduce que la articulación 3 es capaz de soportar hasta 2 Nm sin aumentar bruscamente, simplemente regulando con pequeñas variaciones,

mientras que si el par supera esos 2 Nm, la articulación aumentará su par en 2 Nm con respecto al par que estaba generando antes de aplicar el peso. Esto es; la sensibilidad de la articulación 3 es de 2 Nm.

En esta prueba, los datos se anotaron directamente de pantalla, el error que se puede cometer es de un 5% en la lectura del dato.

6.2.5 PRUEBA CON LA RUTINA "HOME"

En esta prueba se han conectado todos los motores, y medido todos los pares generados por cada articulación.

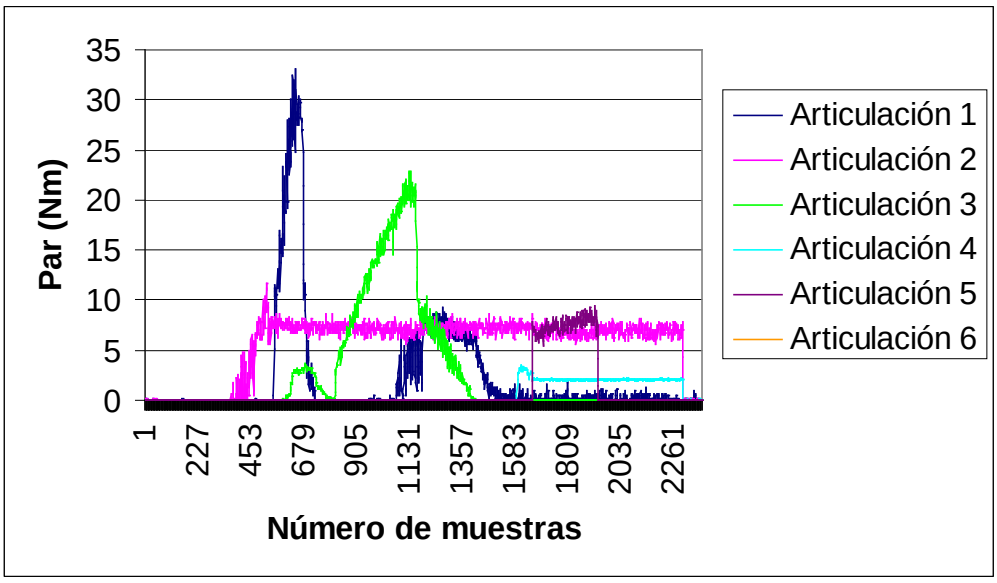

Figura 6.8: Par generado en la rutina home

La rutina de "home" mueve las articulaciones una a una y las sitúa en su posición inicial para, que la odometría sea correcta.

La articulación 2 es la primera en empezar a moverse; una vez que termina su movimiento, se queda enclavada en su posición de "home", ofreciendo una resistencia al movimiento. La articulación 1 es la siguiente en movimiento, esta se eleva un poco frente a la vertical, para luego volver a su posición. Nótese que cuando llega a su posición el motor no genera ningún par. En la articulación 3 se observa, que al elevarse la articulación 1, tiene que generar par para no moverse, pues al ser su eje de giro paralelo al eje de giro de la articulación 1, se ve afectada por su movimiento; lo mismo ocurre cuando es la articulación 3 la que se mueve, la articulación 1 tiene que aguantar

la fuerza que genera el cambio del centro de gravedad del brazo. Tras el movimiento de la articulación 3, la articulación 1 se queda enclavada y excitada para poder soportar la posición de "home" de la articulación 3. Para la articulación 4, como es el giro de antebrazo, solo tendrá que vencer la inercia de las articulaciones 5 y 6, por eso su par es muy pequeño y permanece constante durante el movimiento de las demás articulaciones. La articulación 5, que tiene también un eje paralelo a las articulaciones 3 y 1, al estar en el extremo del brazo, no se ve afectada por los movimientos de las anteriores articulaciones, sin embargo, la articulación 1 si se ve afectada por el movimiento de esta, pues el peso del brazo varía al modificarse la geometría del mismo. También se verá afectada la articulación 3 pero en menor medida que la articulación 1. Cabe destacar que el par que genera el motor 6 es nulo, pues el motor 6 no tiene carga alguna.

6.3 COMPROBACIÓN DE LOS DATOS EN DINÁMICA

La comprobación de los datos obtenidos cuando el motor está en movimiento se realiza calculando la potencia que genera el motor en dicho movimiento, y comparándola con la potencia teórica necesaria para mover esa carga a la velocidad determinada.

Para ello, se obtiene experimentalmente el par generado por el motor; éste será el par de la articulación divido por la relación del harmonic drive. Con este dato, y a través del parámetro Km proporcionado por el fabricante, se puede calcular la potencia eléctrica consumida, mediante la ecuación.

$$W = Nm^2 / Km^2$$ Ecuación 1

W: Potencia eléctrica consumida por el motor.

Nm: Par generado por el motor durante el movimiento.

Km: Constante del motor, 0.098 (Nm / $W^{0.5}$) para el motor 3.

Tras la aplicación de esta ecuación a los datos obtenidos experimentalmente, y teniendo en cuenta sólo el movimiento desde 0º a 90º con respecto a la vertical, se obtiene la siguiente gráfica.

Figura 6.9: Potencia eléctrica consumida

Para hacer la comprobación teórica del movimiento necesitamos calcular la potencia mecánica que tendrá que generar el motor para mover la carga a la velocidad determinada por el movimiento.

Utilizando la siguiente ecuación podemos calcular la potencia mecánica teórica.

$$P = M.\varpi$$ Ecuación 2

P: Potencia mecánica del motor.

M: Par que vence el motor aplicado al eje del motor

ω: Velocidad angular del motor.

Sabiendo que la velocidad del motor es de 500 RPM, que corresponden a 52,35 rad/s, y que el par varía en función del ángulo de la articulación con la vertical, siendo el máximo par para la articulación 3 de 16 Nm a 90º, y dividiendo este dato por la

relación de pares del harmonic drive, el máximo par en el eje del motor será de 0.157 Nm. De estos datos obtenemos la gráfica de la potencia mecánica en función del ángulo de la articulación.

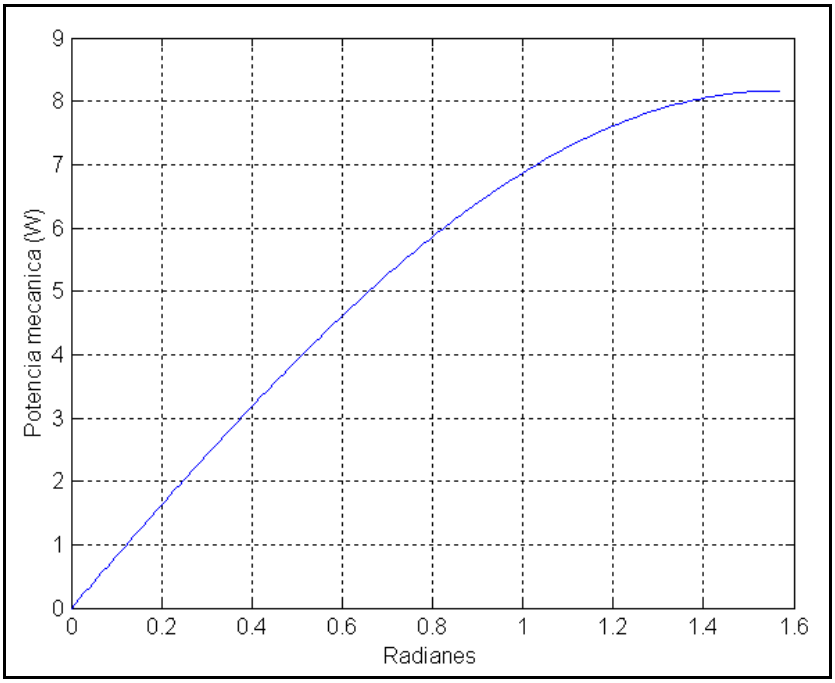

Figura 6.10: Potencia mecánica teórica

Comparando las dos gráficas, se observa que el incremento de potencia a medida que aumenta el ángulo de la articulación con respecto a la vertical es similar, la gráfica teórica tiene su valor máximo de 8.2 W en los 90º de variación con la vertical, mientras que la gráfica obtenida experimentalmente tiene su máximo de 8.8 W en la misma posición de la articulación.

6.4 SIMULACIÓN DEL MOTOR 3 CON MATLAB

En este apartado se pretende justificar los resultados obtenidos experimentalmente con el funcionamiento teórico del motor. Para ello se ha simulado el funcionamiento del motor 3 a través de Matlab, usando la herramienta Simulink. Aunque el motor es un motor brushless, por simplificación y sencillez se va a utilizar el modelo matemático de un motor de DC convencional, ya que su funcionamiento es similar. Se supondrá que no existe reductora a la salida del motor, por lo tanto, a la hora de aplicar la carga se tendrá

en cuenta que el par resistente en el eje del motor será el obtenido experimentalmente, dividido entre la relación de pares del harmonic drive.

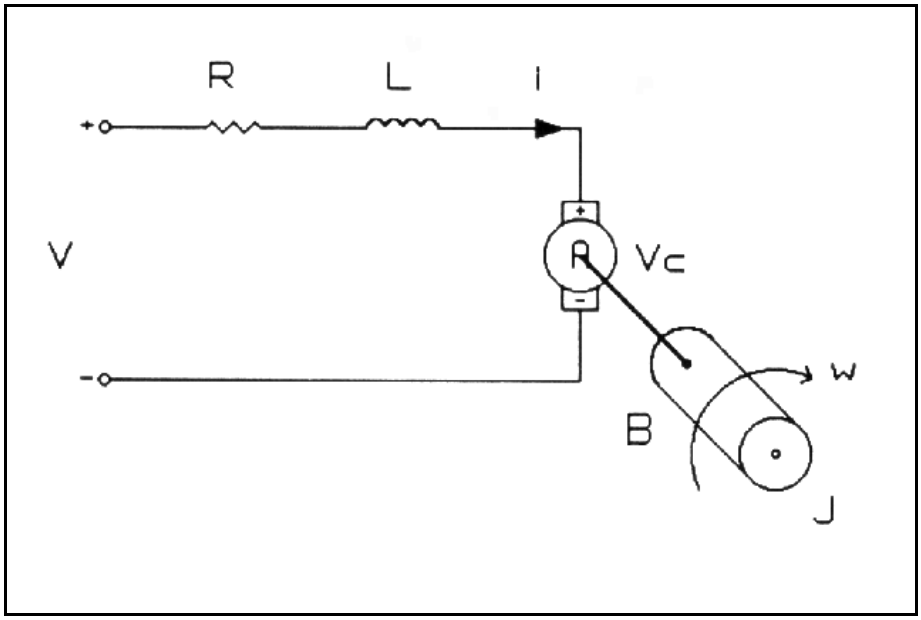

Figura 6.11: Representación eléctrica de un motor DC

V: Tensión de entrada.

R: Resistencia del motor.

L: Inductancia del motor.

I: Corriente que circula por el motor.

Vc: Fuerza contra electromotriz.

J: Momento de inercia del motor.

B: Coeficiente de fricción de la carga.

ω: Velocidad angular de salida.

De las ecuaciones del motor DC convencional, se sabe que el par creado eléctricamente depende directamente de la corriente que circula por él.

$$Tm = Kt.I \qquad \text{Ecuación 3}$$

Siendo *Kt*, la constante motor expresada en (Nm/Amp).

Este mismo par mueve la inercia de la carga así como su resistencia, y si suponemos que la carga es variable.

$$Tm = J\frac{d\varpi}{dt} + B\varpi + Tl$$ Ecuación 4

Tl es la carga referida al eje del motor.

Igualando ambas ecuaciones se obtiene la expresión mecánica del motor, que expresada a través de la transformada de Laplace, y sabiendo que nuestra variable de salida será la velocidad del motor, se obtiene:

$$\varpi(s) = \frac{I(s)Kt - Tl(s)}{Js + B}$$ Ecuación 5

Se necesita otro paso debido a que no podemos variar la corriente si no es a través de tensión de entrada. Del circuito eléctrico del motor obtenemos:

$$V = RI + L\frac{dI}{dt} + Kb.\varpi$$ Ecuación 6

Aplicando la transformada de Laplace y despejando la corriente.

$$I(s) = \frac{V(s) - Kb.\varpi(s)}{Ls + R}$$ Ecuación 7

Expresando las ecuaciones 5 y 7 mediante diagrama de bloques, y sabiendo que las variables de entrada al sistema son la tensión aplicada al motor y la carga, se tiene un diagrama de bloques del motor.

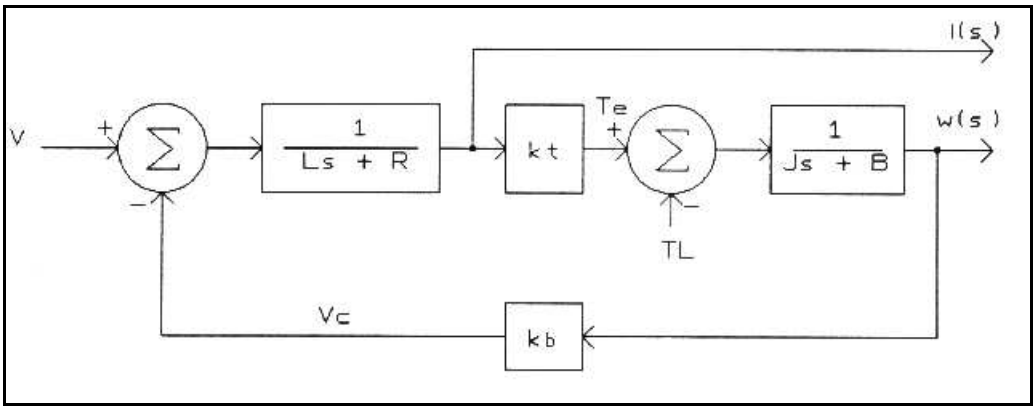

Figura 6.12: Diagrama de bloques de las ecuaciones 5 y 7

Los parámetros para la representación a través de Simulink, serán los dados por el fabricante del motor. Para el motor 3 obtenemos los siguientes parámetros.

Parámetro	**Valor**
R	1.82 Ohms.
L	1.3 mH
Kt	0.132 Nm/A
J	1.98e-5 kgm^2
B	3e-3 Nm/rad/s

Tabla 6.2: Parámetros para la simulación

Para la simulación el parámetro *Kb* tiene el mismo valor que el parámetro *Kt*, cuando se representa en unidades del sistema internacional. Las unidades de *Kb* serán V/rad/s.

El coeficiente de rozamiento de la carga se ha calculado dividiendo la potencia máxima generada por el motor durante el movimiento, (ver figura 6.9), por el cuadrado de la velocidad del eje del motor en movimiento, que como se dijo, es de unas 500 RPM equivalente a 52.35 rad/s; esta operación nos da un valor para *B* de 3e-3 Nm/rad/s.

El coeficiente de rozamiento, varía dependiendo de la carga, pero al estar dentro de una de las funciones de transferencia, resulta complicado simular esa dependencia. Por eso se ha calculado el coeficiente para la situación más desfavorable, que es la posición final, esto nos permite comparar los resultados experimentales con los teóricos en dicha posición.

En la figura 6.13 se observa que la carga es un parámetro de entrada, la carga se representa mediante una fuente senoidal de tensión con una amplitud de 0.15 V y una frecuencia de 0.31 rad/s, para representar la velocidad a la que varía dicha carga. La amplitud se calculó sabiendo el par que soporta el motor cuando la articulación está a 90º con respecto a la vertical. En la figura A.6 se muestra este par máximo, que corresponde a 16 Nm. Si calculamos la carga referida al motor, dividiendo el dato por la relación del harmonic drive, se obtiene un par máximo de 0.15 Nm.

En la siguiente figura se muestra el diagrama usado en Simulink para la simulación del motor.

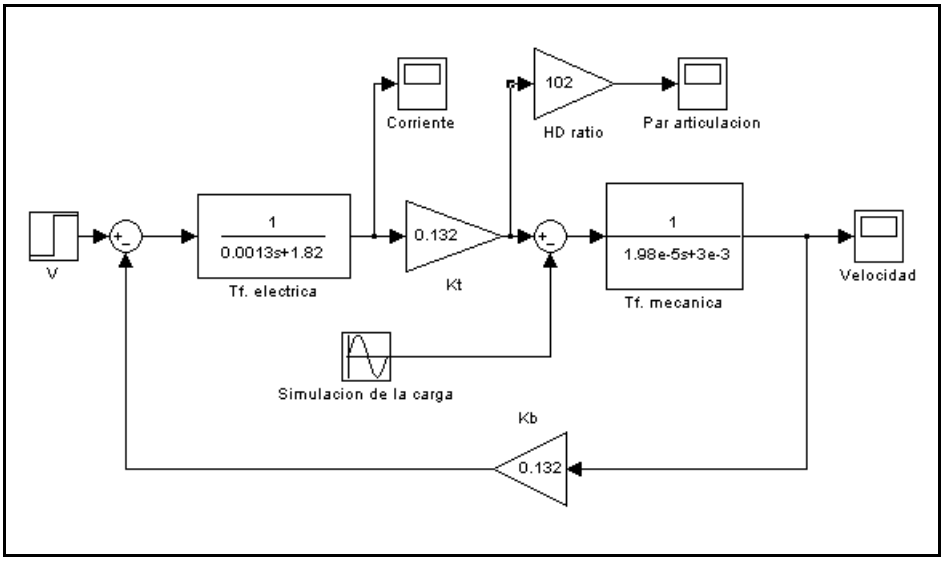

Figura 6.13: Diagrama de bloques usado para la simulación

En este diagrama de bloques se han puesto tres puntos de control de variables, estas son la velocidad angular, la corriente, y el par que genera la articulación.

Para la simulación, la velocidad del eje del motor debe de ser de 52,35 rad/s cuando la carga es máxima, esto implica que habrá que ajustar la tensión de entrada para que el motor tenga esa velocidad, esto es necesario para que la simulación tenga las mismas características que el sistema real en movimiento.

En el cálculo del par de la articulación se utiliza simplemente una ganancia, que corresponde a la relación de pares del harmonic drive.

En la simulación se obtiene los siguientes resultados.

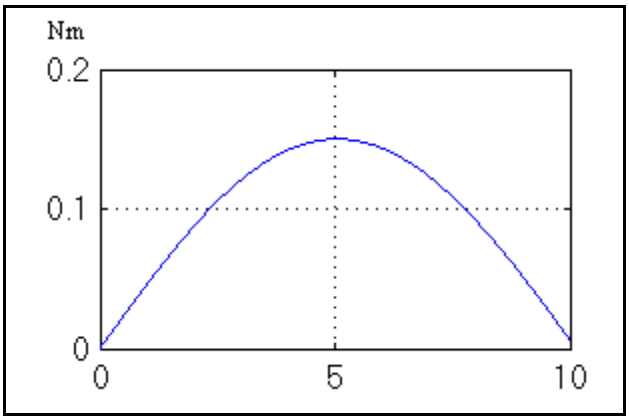

Figura 6.14: Variación de la carga

Esta gráfica muestra la evolución de la carga a medida que va aumentando el ángulo con respecto a la vertical, tiene su máximo a los 5 segundos, que es cuando la articulación llega a la posición horizontal.

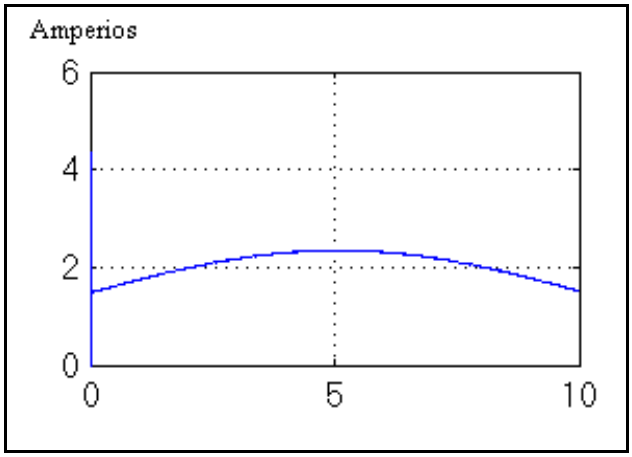

Figura 6.15: Evolución de la corriente

En esta figura se muestra la variación de corriente que circula por el motor, nótese que al principio de la gráfica, su valor aumenta hasta los 4 amperios, este incremento es debido a la inercia; en la realidad ese incremento de corriente es tan rápido que el propio sistema de adquisición no permite ver el efecto. También se destaca que la corriente, una vez vencida la inercia, no empieza en cero, sino que empieza en 1.5 A, este valor se debe a que el coeficiente de rozamiento de la carga no es nulo, como ocurre en la realidad, cuando la articulación se encuentra en la posición inicial.

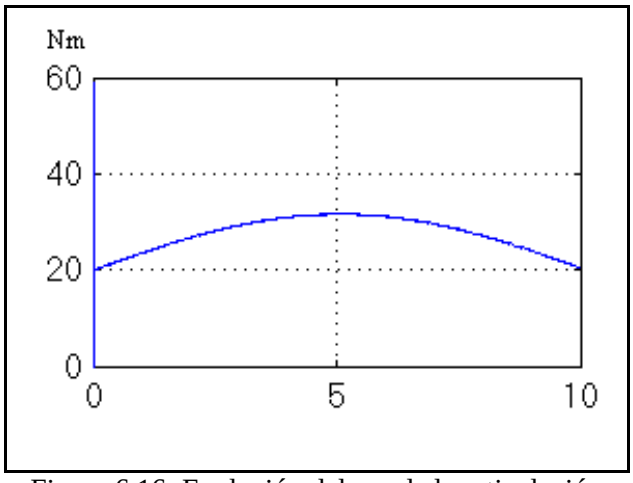

Figura 6.16: Evolución del par de la articulación

En la figura 6.16 se puede observar como varía el par de la articulación cuando varía la carga, llegando hasta los 30 Nm, que es lo mismo que se ha obtenido experimentalmente, a través del USB-DUX.

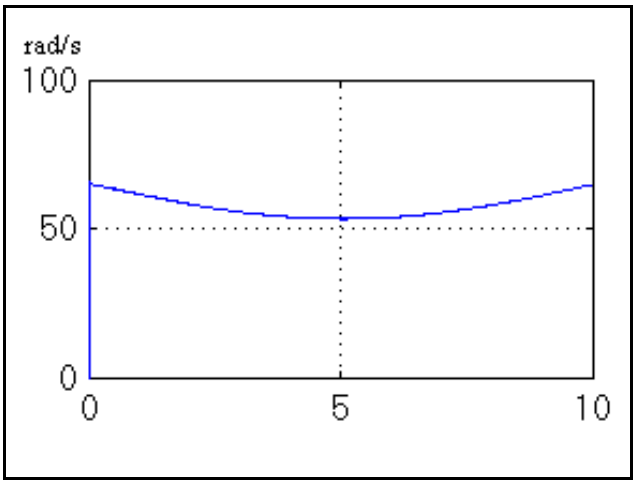

Figura 6.17: Evolución de la velocidad del motor

La figura 6.17 sirve para comprobar que cuando la carga es máxima, la velocidad es la deseada para la simulación; si la velocidad fuese menor, el coeficiente de rozamiento influiría menos y por lo tanto el par generado sería menor; si la velocidad fuese mayor, el coeficiente de rozamiento influiría más y por lo tanto sería mayor el par generado. También se demuestra con esta gráfica que la velocidad va disminuyendo a medida que aumenta la carga.

En la realidad, la tarjeta PMAC aumenta la tensión a medida que aumenta la carga para mantener la velocidad constante. En esta prueba se ha utilizado una tensión de excitación continua, calculada para que cuando la carga sea máxima la velocidad sea la experimental.

CAPÍTULO 7
CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

En este tema, se presentan las explicaciones más detalladas de las pruebas realizadas en el laboratorio con Manfred.

Para poder entender el funcionamiento del motor hay que tener en cuenta que el motor es controlado por la tarjeta PMAC, y que el comportamiento de este dependerá de la programación de dicha tarjeta.

7.1.1 EXPLICACIÓN DE LOS MOVIMIENTOS

Como se dijo en apartados anteriores, el driver del motor esta configurado para trabajar en “open loop”. De esta forma se controlará el motor brushless como si fuera un motor de corriente continua convencional, es decir, mediante tensión. Por lo tanto la tensión aplicada al motor por el driver, será proporcional a la señal que se le comande.

Este modo de trabajar de los drivers tiene unas características singulares. Cuando la carga a la que está sometido el motor es baja, la velocidad estará limitada por la constante denominada “fuerza contraelectromotriz”, que para motores con rotor de imán permanente, es la relación entre la tensión aplicada a las bobinas y la velocidad del motor; mientras que si la velocidad es baja, la corriente que circula por las bobinas está limitada por la resistencia de las mismas.

En todas las pruebas en las que el motor pasa de un estado de movimiento a un estado estático, se observa una bajada instantánea de par, esto es debido a que el motor sólo soporta la carga, sin influencia del coeficiente de rozamiento de ésta. Cuando el motor pasa de un estado estático a un estado de movimiento, hay un aumento de par; este aumento es debido a que tiene que vencer la resistencia estática de la carga.

Al principio de cada una de las pruebas, cuando el motor parte de la situación de reposo, en las gráficas se puede observar un aumento instantáneo del par. Este será mayor cuanto más velocidad se le quiera dar al motor, y será menor cuanto menos velocidad se le dé. La tensión necesaria para obtener la velocidad deseada es comandada a través de la tarjeta PMAC.

Cuando el motor está en movimiento, el par que genera la articulación va aumentando a medida que lo hace también el peso del brazo para la articulación en movimiento, para lograr que la velocidad permanezca constante, hay que aumentar el

par a medida que aumenta el peso. Cada uno de los motores cuenta con un enconder, solidario al eje del motor, que proporcionan la medida de velocidad y posición a la tarjeta PMAC que los controla; esta tarjeta, detecta a través de los encoders, la disminución de velocidad que se produce debido al aumento de peso, y para mantener la velocidad constante, aumenta la tensión del motor, que debido al modo de funcionamiento, aumentará su par y su velocidad hasta adecuarla con la fijada previamente.

Habrá que tener en cuenta que la tarjeta PMAC no proporciona todo tipo de pares en movimiento, esta tarjeta tiene unos límites programados para no dañar a los motores; si en algún momento, el par necesario para mantener la velocidad preprogramada es superior a esos límites, el motor dejará de moverse y se quedará enclavado en esa posición. Este efecto se puede observar en la prueba número 3, en la figura 6.5.

Cuando el motor se encuentra en estática, el par puede aumentarse sin necesidad de mover el motor. Cuando la tarjeta está en configuración de paro de motor, y el encoder detecta que hay un pequeño movimiento, la tarjeta PMAC actuará en consecuencia, aumentando o disminuyendo la tensión para aumentar o disminuir el par. En la prueba 4, en las figuras 6.6 y 6.7, se puede observar que la tarjeta PMAC actúa incrementando el par poco a poco conforme la fuerza externa aumenta, hasta que detecta una variación demasiado grande con respecto a la posición inicial, y aumenta bruscamente el par para que no se produzcan más variaciones; esta situación se repetirá hasta que se llegue al límite de par en estática del motor, pues si se supera esta situación, el motor puede sufrir daños.

Cuando la fuerza aplicada es en contra de la gravedad, el control es distinto, el motor disminuirá el par hasta que la fuerza aplicada iguale la fuerza debida al peso del brazo, y a partir de aquí, se comportará como se ha explicado antes. Irá aumentando el par para mantener la articulación en la posición en la que se paró. Se destaca que cuando el motor es sometido a una fuerza externa a favor de la gravedad, y después se deja de aplicar dicha fuerza, la respuesta a esa variación negativa de par, será más limitada.

7.1.2 RUIDO

En todas las pruebas, se observa que los datos obtenidos, cuando el motor está en estática, tienen una componente de ruido que provoca que los datos varíen en torno al 5%, estas pequeñas variaciones no son debidas al ruido que genera el USB-DUX sino que es debido al ruido generado en el propio driver. Cuando la articulación está en movimiento se observa que este ruido aumenta en algunas lecturas, llegando algunas incluso a variar hasta el 15%, esta variación es producida por el instante de adquisición del USB-DUX. Al ser un motor brushless, la tensión tiene que ser proporcionada siguiendo la secuencia de fases que marca el fabricante del motor, si mientras el sistema de adquisición hace el muestreo, el driver está haciendo el cambio de fases, puede que la lectura obtenida sea menor de la esperada. Para disminuir este error se ha introducido dentro del código fuente del programa un pequeño filtro paso bajo.

7.1.3 VELOCIDAD MÁXIMA DE ADQUISICÓN

Experimentalmente se comprueba que el USB-DUX funciona con una velocidad de adquisición de 96 datos por segundo. Debido a que obtienen datos de 6 canales diferentes, de un canal tomará muestras a una velocidad de 16 datos por segundo. Puesto que la variación del par en los motores es una variación lenta, no se necesitan grandes velocidades de adquisición. La velocidad de adquisición depende mucho de la velocidad del microprocesador del ordenador y de las tareas que esté ejecutando en ese momento, ya que no se le hace trabajar con la característica de tiempo real de Linux, la cual, da más prioridad a las interrupciones hardware.

7.2 TRABAJOS FUTUROS

Tras la realización del proyecto surgen varias ideas para trabajos futuros.

Al igual que se obtiene el par de cada articulación del brazo, y sabiendo que el USB-DUX cuenta con ocho entradas analógicas, se pueden usar las dos entradas no utilizadas para la obtención del par generado por los motores de la base móvil. Esta obtención seria muy útil a la hora de saber si Manfred se desplaza por un plano

inclinado, y si podrá soportar dicha inclinación, también sería eficaz para saber si las ruedas están patinando. Estos motores funcionan igual que los del brazo, por lo tanto sólo habrá que cambiar el programa para que obtenga datos de todo los canales.

Ahora, al controlar el par instantáneo que genera cada motor, se puede controlar el brazo en par. Este control es muy deseable, ya que Manfred está diseñado para trabajar en entornos humanos. Si se detecta un incremento de par en un movimiento, se puede interpretar como un obstáculo que se ha interpuesto en el camino, y parar el motor para evitar daños.

ANEXO A
CÁLCULO DE LOS PARES SOPORTADOS POR CADA ARTICULACIÓN

En este anexo se pretende calcular el par creado por el peso del brazo; para ello, habrá que tener en cuenta los diagramas de fuerza aplicados en el brazo de Manfred. Tal cual esta situado el brazo, podemos asemejar su diagrama de fuerzas al de una partícula solidaria a una barra rígida sometida a la fuerza de la gravedad, es decir, un péndulo rígido como el que se muestra en la figura.

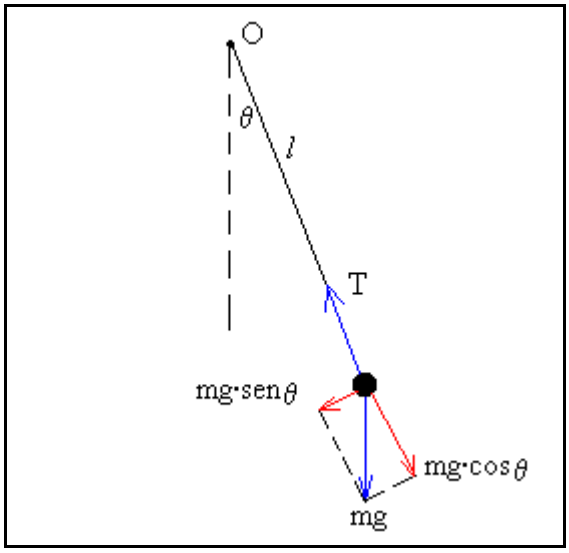

Figura A.1: Analogía del brazo con un péndulo

Aunque en la figura se muestra el diagrama de una sola partícula, sirve como ejemplo para el brazo de Manfred. Cada motor será tratado como una partícula diferente aplicada a una distancia del eje de giro diferente. Como se observa en la figura anterior, sólo nos generará par la fuera tangencial, y esta dependerá del ángulo en el que se encuentre el brazo, por lo tanto, la ecuación para calcular el par debido al peso del brazo es:

$$\tau_j = \sum_i (P_i.d_i).g.sen(\theta(t)) \qquad \text{Ecuación 1}$$

P_i : peso del motor con su harmonic drive y herrajes de sujeción.

d_i : distancia entre el eje de giro del motor *j* y P_i.

g: aceleración de la gravedad (9.8 m/s^2).

θ(t): ángulo de posición del brazo con respecto a la vertical.

Para calcular el par que tendrá que soportar cada articulación se tiene en cuenta el peso de los motores con los harmonic drives y sus herrajes de sujeción, y la distancia a la que se encuentran de la articulación que soportará dicho par. El peso de los eslabones de fibra de carbono también se considera en el cálculo, pero la distancia será la tomada desde el centro de gravedad de dicho eslabón hasta el eje de giro de la articulación para la que se calcula dicho par.

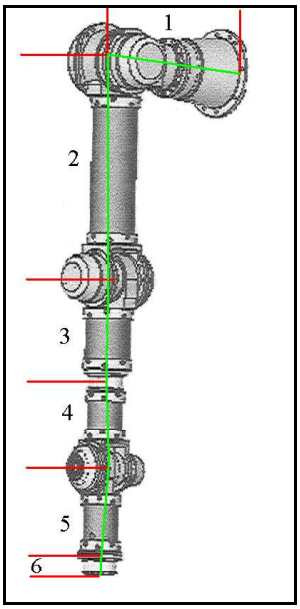

Figura A.2: Eslabones del brazo de Manfred

En esta figura se indican los eslabones del brazo móvil, y en la siguiente tabla, la longitud y el peso de cada uno, para el cálculo de los distintos pares.

Eslabón	**Longitud (mm)**	**Peso (Kg)**
1	250	----
2	400	0.85
3	187	0.54
4	163	0.24
5	178	0.35
6	50	----

Tabla A.1: Longitud y peso de los eslabones

El brazo tiene un sensor de fuerza-par en el extremo de la articulación 6, con un peso aproximado de 200 gramos, habrá que tener esto en cuenta a la hora de calcular el par. Este sensor no está incluido en la figura A.2.

En la siguiente tabla se expresa el peso de cada motor junto con el harmonic drive, el peso de sus herrajes está separado.

Articulación	**Peso de motor + HD (Kg)**	**Peso herrajes (Kg)**
1	2.8	0.997
2	2.8	1.36
3	1.62	0.88
4	0.984	0.144
5	0.984	0.482
6	0.944	0.133
7 (Sensor fuerza-par)	0.2	----

Tabla A.2: Peso de cada articulación

Con todos estos datos, y teniendo en cuenta que el peso total es la suma de los pesos aquí descritos, se puede pasar a calcular el par máximo soportado por cada articulación.

La ecuación general del cálculo de par es:

$$\tau_j = \sum_{i=j+1}^{7} (P_i . d_{i-1}) g \qquad \text{Ecuación 2}$$

Esta ecuación expresa el par máximo que deben soportar las articulaciones. Este par se da cuando el motor ha movido dicha articulación hasta formar un ángulo de 90º con la vertical.

A.1 CÁLCULO DEL PAR SOPORTADO POR CADA ARTICULACIÓN

Para la articulación 1, observamos que el motor 2 esta dentro del eje de giro de dicha articulación, por lo tanto, el par que provocará el motor 2 con respecto a la articulación 1 será nulo, además se observa que la articulación 2, por emplazamiento y eje de giro, soportará el mismo par que el anterior, por lo tanto, para calcular el par soportado por las articulaciones 1 y 2 habrá que tener en cuenta los pesos de los motores 3, 4, 5, 6 y sensor de fuerza-par, con sus respectivas distancias al centro del eje del motor 1.

Para la articulación 1 y 2:

Articulación	Distancia (m)	Peso (Kg)	Par (Nm)
3	0.4	2.5	9.8
4	0.589	1.12	6.46
5	0.752	1.46	10.68
6	0.93	1.07	9.8
7, Sensor fuerza-par	0.98	0.2	1.92

Tabla A.3: Pares aplicados a los motores 1 y 2 (a)

Para saber el par en Nm que se le aplica, simplemente hay que sumar todo los pares debidos a cada motor, por lo tanto, se obtiene que están soportando un par de 38.66 Nm. Antes de tomar estos resultados como definitivos, hay que calcular el par debido a los eslabones.

Eslabón	Distancia del CDM al eje (m)	Peso (Kg)	Par (Nm)
2	0.2	0.85	1.67
3	0.49	0.54	2.6
4	0.66	0.24	1.55
5	0.84	0.35	2.88

Tabla A.4: Pares aplicados a los motores 1 y 2 (b)

Sumando todos los pares obtenemos un resultado de 8.7 Nm, ahora bien, si añadimos este resultado al valor de 38.66 Nm calculado anteriormente, se tiene que el par soportado por las articulaciones 1 y 2 es de 47.36 Nm. Por lo tanto podemos concluir que para dichas articulaciones, el par que soportan en la condición más desfavorable es de 47 Nm.

Para la articulación 3, haciéndolo de forma análoga:

Articulación	**Distancia (m)**	**Peso (Kg)**	**Par (Nm)**
4	0.187	1.12	2.04
5	0.35	1.46	5
6	0.528	1.07	5.53
7, Sensor fuerza-par	0.578	0.2	1.13

Tabla A.5: Pares aplicados al motor 3 (a)

Sumando todos los pares, obtenemos un par de 13.7 Nm. Ahora se calculan los pares debidos a los eslabones.

Eslabón	**Distancia del CDM al eje (m)**	**Peso (Kg)**	**Par (Nm)**
3	0.093	0.54	0.49
4	0.27	0.24	0.63
5	0.44	0.35	1.5

Tabla A.6: Pares aplicados al motor 3 (b)

Y finalmente sumando todos los pares obtenidos, se obtiene que la articulación 3 soporta 16.32 Nm. Para el cálculo del rendimiento se aproxima a 16 Nm.

Observando la disposición y el movimiento de los motores 4 y 6 se puede deducir que estas articulaciones no soportarán par alguno en condiciones normales, son motores sin carga, pues su eje de giro atraviesa el centro de la carga, el par de estos motores sólo se verá afectado por la inercia de rotación de esta.

La articulación 5, aunque soporta un par muy pequeño y se puede despreciar para el cálculo del rendimiento de los harmonic drives, también se calcula de la misma forma.

Articulación	**Distancia (m)**	**Peso (Kg)**	**Par (Nm)**
6	0.178	1.07	1.86
7, Sensor fuerza-par	0.228	0.2	0.44

Tabla A.7: Pares aplicados al motor 5 (a)

Calculando el par creado por el eslabón.

Eslabón	**Distancia del CDM al eje (m)**	**Peso (Kg)**	**Par (Nm)**
6	0.089	0.35	0.3

Tabla A.8: Pares aplicados al motor 5 (b)

Sumando todos los pares calculados por separado para esta articulación, resulta que soportará un par de 2.6 Nm. Es un par que se puede despreciar para el cálculo del rendimiento del harmonic drive.

Los pares anteriormente calculados, parten de la premisa que sólo se mueve una articulación y que las demás permanecen inmóviles y estiradas; dicho par sólo será soportado cuando la articulación llegue a estar a 90º con respecto a la vertical.

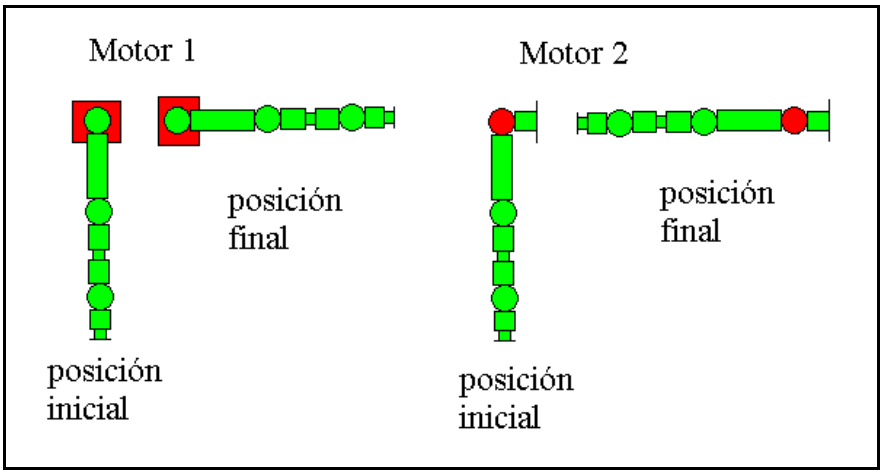

Figura A.3: Movimientos debidos a los motores 1 y 2

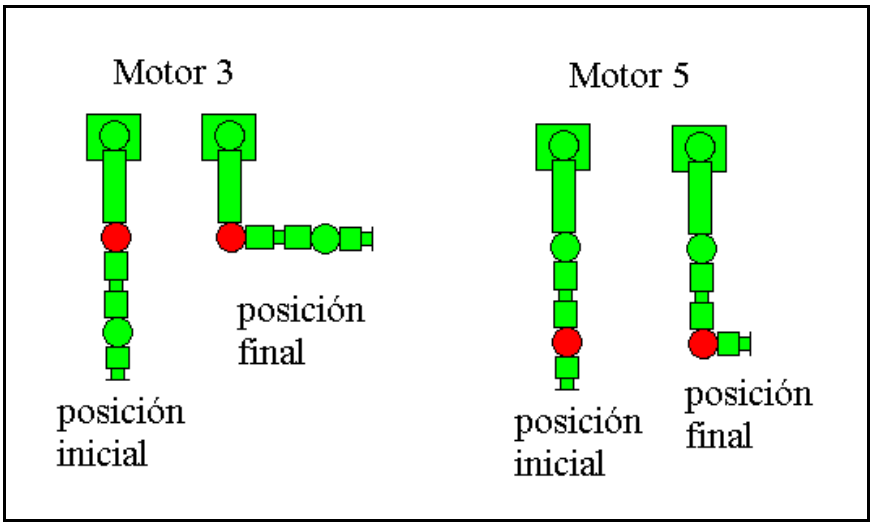

Figura A.4: Movimientos debidos a los motores 3 y 5

En las Figuras anteriores se muestran los movimientos para los que están calculados los pares, marcando en rojo las articulaciones que están soportando el par.

El par máximo, calculado anteriormente, será soportado en la posición que se ha denominado "posición final", mientas que en posiciones intermedias, el par que soportaran vendrá influenciado por el ángulo que tenga la articulación con respecto a la vertical, como se demuestra en la ecuación 1.

Las siguientes figuras muestran, de forma teórica, como evoluciona el par estático que soportan las articulaciones con respecto al ángulo que forman con al vertical.

Para las articulaciones 1 y 2

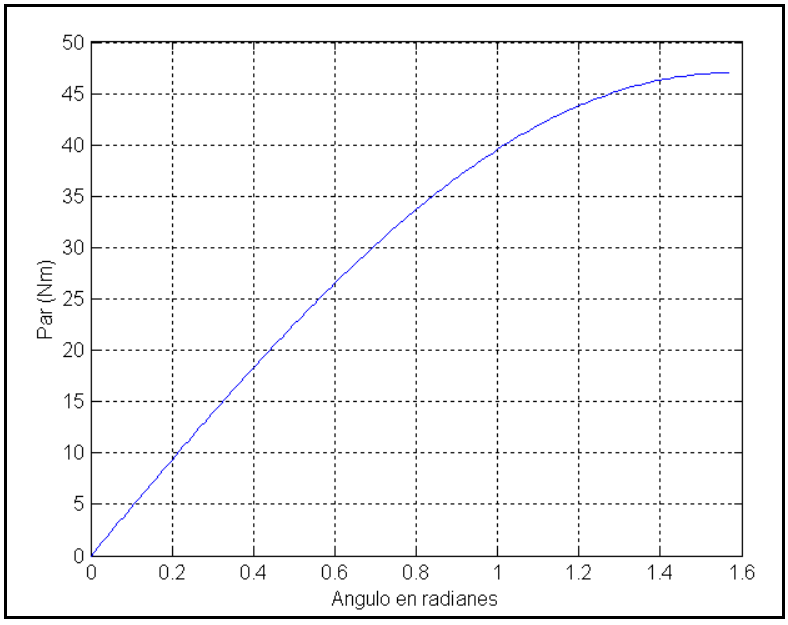

Figura A.5: Evolución del par estático soportado por las articulaciones 1 y 2

Para la articulación 3

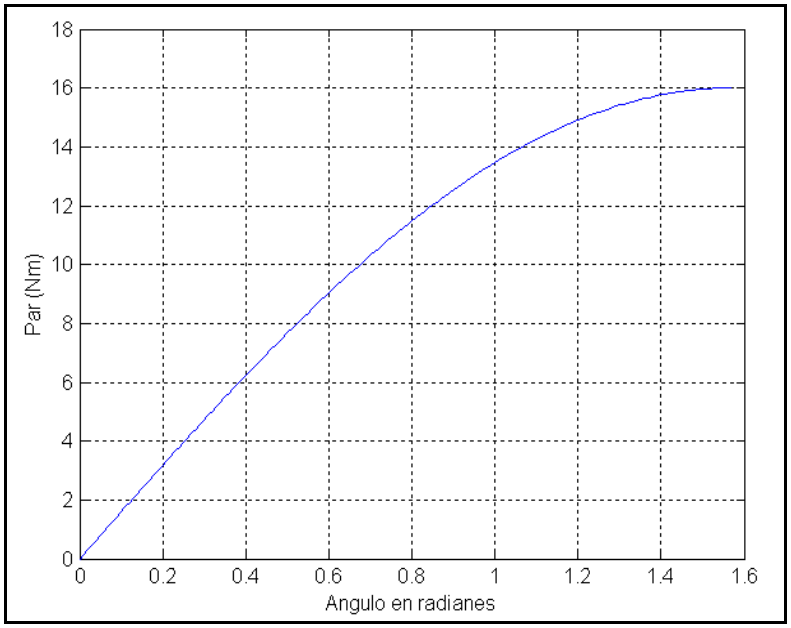

Figura A.6: Evolución del par estático soportado por la articulación 3

Para la articulación 5

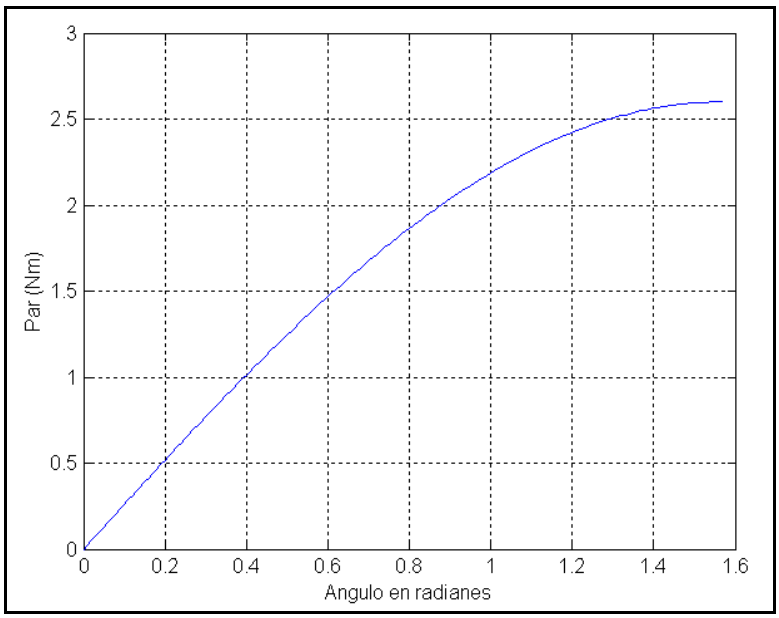

Figura A.7: Evolución del par estático soportado por la articulación 5

ANEXO B
CÁLCULO DEL RENDIMIENTO DE LOS HARMONIC DRIVES

El harmonic drive es una de las pocas transmisiones que pueden ser reversibles, es decir, el motor, que normalmente es el que aplica el movimiento, también puede usarse como freno; esta característica ayuda a calcular la respuesta del motor cuando se le aplica una carga externa.

El rendimiento del harmonic drive depende de: La velocidad de entrada de la transmisión, de la temperatura a la que se trabaja, del tipo de lubricación usado y de la carga aplicada. Todo esto tendremos que tenerlo en cuenta a la hora de calcular dicho rendimiento.

El fabricante proporciona unas gráficas con el rendimiento del harmonic drive en función de los parámetros expresados anteriormente.

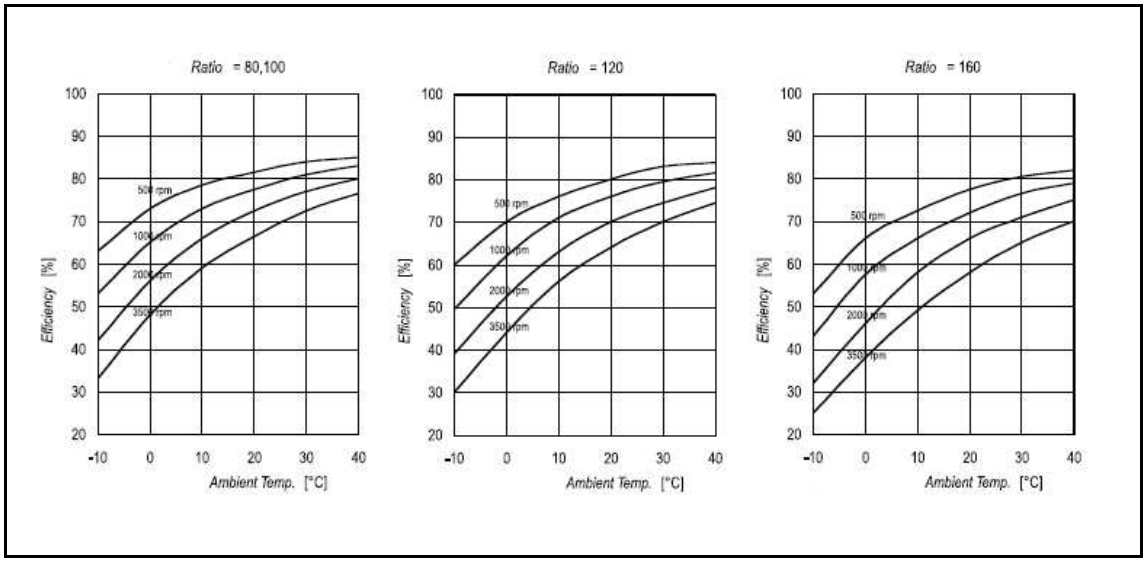

Figura B.1: Eficiencia nominal de los harmonic drives

La velocidad de entrada se calcula de forma experimental de la siguiente forma:

- Sabemos, por las pruebas realizadas, que el brazo tarda en girar 90º aproximadamente 5 segundos, por lo que dará una vuelta cada 20 segundos.
- Si cada 20 segundos gira una vuelta, se deduce que dará 3 vueltas en un minuto.
- Como la relación de velocidades del harmonic drive del motor 3 es de 160, obtenemos que el motor girará a unas 480 revoluciones por minuto, que podemos aproximar a 500 rpm.

Para la temperatura de trabajo, basta con elegir la temperatura normal de 20ºC.

En la lubricación de los harmonic drives del brazo se usa grasa.

Ahora bien, una vez que se obtiene el rendimiento nominal de cada harmonic drive a través de la graficas de la figura B.1, se pasa a calcular el rendimiento real, pues este variará dependiendo del par de salida, debido al peso del brazo.

Para eso necesitamos parámetros como el par máximo nominal de la transmisión y el par de salida que soporta en ese momento.

En la siguiente tabla se muestran esos datos.

Número de articulación	**Tn (Par medido a 2000 rpm) (Nm)**	**Rendimiento nominal**	**Ta (Par de salida debido al peso del brazo) (Nm)**
1, 2	67	78%	47
3	40	78%	16
4	24	80%	----
5	24	80%	----
6	24	81%	----

Tabla B.1: Parámetros para el cálculo del rendimiento.

Como se observa en la tabla B.1, el par debido al peso del brazo sólo afecta a las articulaciones 1, 2 y 3, por lo tanto el rendimiento de estas ha de ser calculado.

Mención especial requieren las articulaciones 4, 5 y 6, pues al estar sometidas a bajo par de trabajo, el rendimiento que se utiliza para estas es el nominal y no hará falta calcular el rendimiento para el programa de adquisición.

B.1 PROCEDIMENTO DE CÁLCULO

Para calcular el rendimiento se necesita calcular el factor de par, este viene determinado por la expresión:

$$V = \frac{Ta}{Tn}$$ Ecuación 1

Ta: Par de salida debido al peso del brazo

Tn: Par nominal medido a 2000 rpm (dato de fabricante)

V: Factor de par

Una vez calculado el factor de par, se pasa a calcular de forma gráfica el factor de rendimiento ***K***.

Para ello el fabricante nos proporciona una gráfica que relaciona el factor de par con el factor de rendimiento.

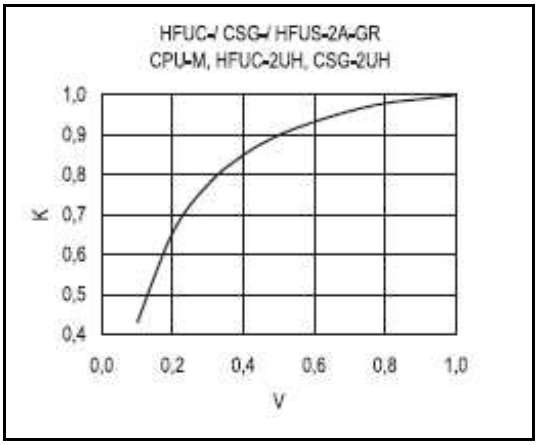

Figura B.2: Relación entre K y V

Con el valor de ***V*** obtenido anteriormente y con la gráfica de la figura B.2, se obtiene el factor ***K***, que relaciona el rendimiento nominal con el rendimiento en función de la carga, de la siguiente forma.

$$\eta_L = \eta.K$$ Ecuación 2

η_L***:*** Rendimiento debido a la carga.

η***:*** Rendimiento nominal del harmonic drive.

K: Factor de rendimiento.

De esta forma se obtienen los rendimientos en función de la carga en cada articulación.

Articulación	**Rendimiento debido a la carga**
1, 2	74%
3	64%

Tabla B.2: Rendimiento debido a la carga

La articulación 4 y 5 tienen un rendimiento de un 80%, mientras que la articulación 6 tiene un rendimiento del 81%.

Para calcular la relación de pares de cada harmonic drive, basta sólo con multiplicar el rendimiento de la transmisión con su relación de velocidades.

ANEXO C
DIAGRAMA DE FLUJO Y CÓDIGO FUENTE DEL PROGRAMA DE ADQUISICIÓN DE DATOS

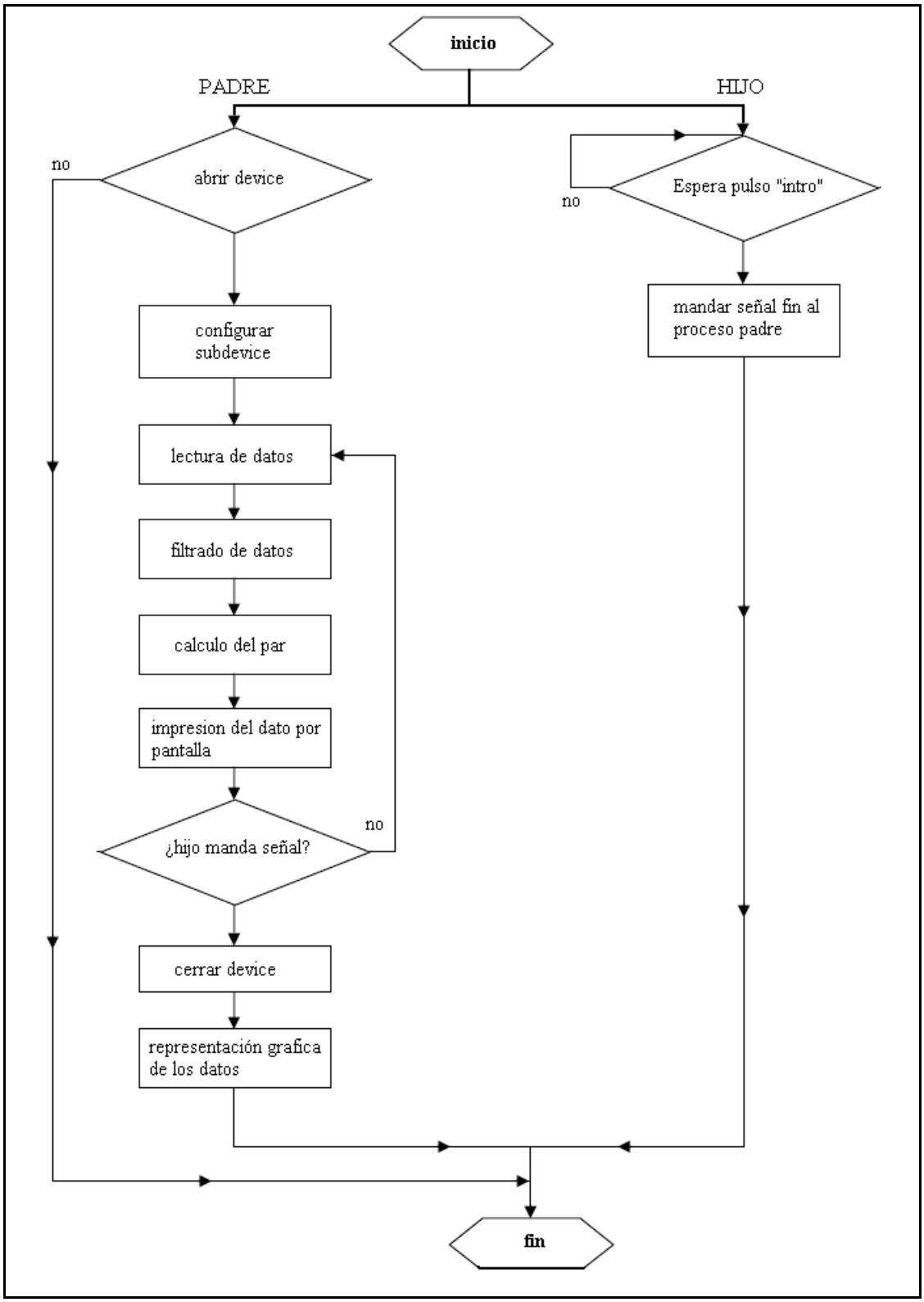

Figura C.1: Diagrama de flujo del programa de adquisición

CÓDIGO FUENTE

```
#include<stdio.h>
#include<comedilib.h>                    /*librería de funciones de Comedi*/
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>
#include<signal.h>

/*DECLARACIÓN DE MACROS Y CONSTANTES*/

#define Km12  0.477*160*0.74
#define Km3  0.132*160*0.64
#define Km45  0.0611*120*0.8
#define Km6  0.0458*100*0.81

#define consumo(inten) 4*inten
#define par(ibobina,constante) ibobina*constante

/*DECLARACIÓN DE FUNCIONES QUE SE UTILIZAN*/

void padre(void);
void tratosig(int s);

/*DECLARACIÓN DE VARIABLES GLOBALES NECESARIAS*/

int b=0;          /*variable para el trato de la señal del hijo*/

/*FUNCION PRINCIPAL*/

int main(void)
{
  pid_t pid;
  int status;
  puts("\n");
  puts("INICIO DE LA OBTENCION DE DATOS");
  puts("Para salir pulse en cualquier momento (Control - d) o (intro)");
  puts("\n");

  switch(pid=fork())
    {
    case(pid_t)-1:
      perror("fork");
      exit(EXIT_FAILURE);

    case(pid_t)0:
      getchar( );
      kill(getppid( ),SIGCHLD);
      break;
```

/*constantes para el cálculo del par, (Km, relación del HD y rendimiento*/

/*código del proceso hijo*/

```
   default:
     padre( );
     wait(&status);
     break;
   }
 exit(EXIT_SUCCESS);
}
```

/*FUNCIÓN DEL PROCESO PADRE*/

```
void padre(void)
{
 int subdev = 0;
 int canal = 0;
 int rango = 2;
 int referen = AREF_GROUND;

 int valores=0;
 comedi_t *cosa;
 lsampl_t dato[6];
 lsampl_t datoant[6];
 lsampl_t dato2[6];
 int maxdato;
 comedi_range *tiporango;
 double voltios[6];
 double parmotor[6];
 double corriente[6];
 FILE *desc;

 cosa=comedi_open("/dev/comedi0");
 if(!cosa){
  comedi_perror("/dev/comedi0");
  exit(0);
 }

 desc=fopen("./motores", "w");
 if (desc==NULL){
  puts("no se puede crear el archivo");
 }

 printf("MOTOR1\tMOTOR2\tMOTOR3\tMOTOR4\tMOTOR5\tMOTOR6\n");

 signal(SIGCHLD,tratosig);

 maxdato=comedi_get_maxdata(cosa,subdev,canal);
 tiporango=comedi_get_range(cosa,subdev,canal,rango);
```

/*definición de variables para Comedi*/

/*definición de variables para la adquisición/

/*apertura del dispositivo*/

/*funciones para la conversión del dato*/

/*bucle infinito para la obtención de los datos*/

```
while(b==0){
   for(canal=0;canal<6;canal++){

     comedi_data_read(cosa,subdev,canal,rango,referen, & dato[canal]);

     if(valores!=0){
          dato2[canal]=dato[canal];
          dato[canal]=(dato2[canal]+datoant[canal])/2;
          datoant[canal]=dato2[canal];
     }
     else{
          datoant[canal]=dato[canal];
     }

     if(dato[canal]<=0){
          printf("%d\t",0);
          fprintf(desc,"%d\t",0);}
     else if(dato[canal]>=4000){
          printf("out\t");
          fprintf(desc,"out\t");}
     else{
```

/*filtro paso bajo*/

/*definición del rango que se representará*/

/*conversión de los datos a unidades físicas*/

```
          voltios[canal]=comedi_to_phys(dato[canal],tiporango,maxdato);
          corriente[canal]=consumo(voltios[canal]);

          if(canal<=1){
            parmotor[canal]=par(corriente[canal],Km12);
          }
          else if(canal==2){
            parmotor[canal]=par(corriente[canal],Km3);
          }
          else if(canal>=3 && canal<=4){
            parmotor[canal]=par(corriente[canal],Km45);
          }
          else{
            parmotor[canal]=par(corriente[canal],Km6);
          }
       printf("%.3f\t",parmotor[canal]);
          fprintf(desc,"%.3f\t",parmotor[canal]);
     }
   }
   printf("\n");
   fprintf(desc,"\n");
   canal = 0;
   valores++;
 }                    /*fin del bucle de adquisición*/
```

/*cálculo del par para cada articulación*/

```
  puts("Programa terminado por el usuario");
  comedi_close(cosa);
  fclose(desc);
  puts("USB-DUX finalizado");
  printf("%d lecturas obtenidas\n",valores);
```

/*representación gráfica de los datos*/

```
 if(execlp("gnuplot","gnuplot","-persist","grafica",NULL)==-1){
    perror("execlp");}
 return;
}
```

/*FUNCIÓN DE TRATO DE SEÑAL*/

```
void tratosig(int s)
{
  b=1;
  return;
}
```

ANEXO D
INSTALACIÓN DE COMEDI EN RED HAT 9

Comedi es compatible con Linux, por lo que se puede instalar en cualquier distribución de Linux, ya sea Debian, Mandrake, Suse, Red Hat, etc.

En la mayoría de distribuciones de Linux, Comedi es completamente compatible con la Kernel y no se necesita ejecutar comandos especiales, solo habrá que seguir las instrucciones que se dan en el apartado D.2; así mismo, existe un pequeño manual de instalación dentro del paquete en que se distribuye Comedi.

Debido a que la Kernel que se distribuye con Red Hat a sido fuertemente modificada, hay que hacer unos pasos previos a la instalación para asegurarse que la compilación de Comedi es satisfactoria. Este proyecto trabaja con Red Hat 9 y con la versión de la Kernel 2.4.20. Todos los pasos que se mencionan aquí han sido probados y funcionan.

Cabe destacar que antes de hacer cualquier instalación de Comedi, surge la necesidad de tener una Kernel compatible con nuestros requerimientos. Como el dispositivo a utilizar para la obtención de datos es el denominado USB-DUX, se necesita que la Kernel tenga compatibilidad total con los dispositivos USB; para ello en la configuración de la Kernel deberán estar marcados las siguientes opciones:

- Support for hot-pluggable devices.
- <Module> Support for USB.
- Preliminary USB device filesystem.
- <Module> UHCI (IntelPIIX4, VIA,...) support.
- <Module> OHCI (Compaq, iMacs, OPTi, SiS, ALi,...) support.
- <Module> EHCI HCD (USB 2.0) support (EXPERIMENTAL).

Después de haber marcado estos requisitos se debe compilar la Kernel.

D.1 HACIENDO LOS ARCHIVOS APROPIADOS PARA LA KERNEL

Esta sección indica paso a paso como se debe hacer la correcta creación de los archivos necesarios en Red Hat.

Primero se necesita tener el código fuente de la Kernel instalada en Red Hat, para ello existe un paquete dentro de la instalación de Red Hat que permite instalar el código fuente. Si se tiene ya esta parte habrá, que pasar a la creación de una copia de los archivos a un directorio especifico.

Para la realización de este procedimiento es necesario tener privilegios de súper usuario.

#cd /usr/src

Nos ponemos en ese directorio y creamos la copia.

#ln –s linux-2.4 linux

Ahora se ha creado una carpeta dentro de ese directorio que se llama “linux”.

Nos vamos al directorio de las configuraciones y copiamos el archivo de configuración.

#cd /usr/src/linux/configs

Copiamos el archivo de configuración necesario para el ordenador. Para este proyecto es el siguiente.

#cp –p kernel-2.4.20-i586.config ../.config

Asegurarse de poner el punto en “.config”

Se borran todas las dependencias de la Kernel, para eso habrá que ir al siguiente directorio.

#cd /usr/src/linux

Nos aseguramos que se ha creado bien el archivo “.config” y borramos dependencias.

#make mrproper

Rehacemos la configuración para la Kernel.

#make oldconfig

Ahora habrá que editar el archivo “Makefile”.

#vim –v Makefile

Cerca del principio del archivo se encuentra la línea.

EXTRAVERSIÓN = -8custom

Solamente habrá que eliminar el bit “custom” y dejarlo como la siguiente línea.

EXTRAVERSIÓN = -8

Guardar y salir del editor.

Se crean las dependencias nuevas.

#make dep

Después de haber hecho todo este procedimiento se debe reiniciar le ordenador y actualizar la herramienta llamada “swig” a la versión 1.3.X o superior. Es necesario para la compilación del paquete “comedilib”. Una vez actualizado se reinicia otra vez el ordenador.

Los archivos y modificaciones pueden ser ligeramente diferentes dependiendo de la versión.

Una vez que se han hecho todos estos pasos se puede pasar a la compilación de Comedi.

D.2 COMPILANDO EL PAQUETE COMEDI

Primero se debe descargar el paquete Comedi, el archivo se llamará “comedi.tar.gz”, aunque puede variar el nombre respecto a la versión de dicho archivo.

Debemos asegurarnos que para la instalación se tienen permisos de administrador y que se copia este archivo a una carpeta en la que se tenga permiso de escritura, una carpeta valida puede ser “/usr/local”.

#cd /usr/local

Ahora habrá que descomprimir el archivo que se ha descargado, para ello se ejecuta:

#tar -xzvf comedi.tar.gz

Una vez se haya descomprimido el archivo, se habrá creado una carpeta nueva llamada “comedi”, ahora vamos a la carpeta creada.

#cd comedi

Una vez en este directorio se necesita configurar la instalación de Comedi, para ello se utilizará el comando.

#./configure

Ahora se crearán los archivos del dispositivo para acceder al hardware desde los procesos de usuario.

#make dev

Este comando crea unos archivos especiales para los dispositivos Comedi, estos archivos son: (/dev/comedi0, /dev/comedi1, /dev/comedi2, etc).

La siguiente operación a realizar es la compilación de Comedi mediante la siguiente instrucción.

#make

Una vez compilado ya podemos instalar los módulos de la Kernel.

#make install

Si se han seguido todos estos pasos y no se ha obtenido ningún error ya tenemos instalado los módulos de Comedi en la Kernel.

D.3 COMPILANDO EL PAQUETE COMEDILIB

La instalación de Comedilib es similar a la de Comedi. Existe un pequeño manual de instalación dentro de este paquete. Comedilib se puede descargar de la misma pagina que Comedi; también se necesita tener permisos de administrador y tener el archivo copiado en la carpeta "/usr/local".

#cd /usr/local

#tar -xzvf comedilib.tar.gz

#cd comedilib

Hasta esta parte hemos descomprimido el archivo en "/usr/local/comedilib" para poder instalarlo

#./configure --sysconfdir=/etc

Configuramos Comedilib y decimos al programa que el directorio de configuración se encuentra en la carpeta "/etc".

#make

Compila Comedilib para su posterior instalación.

#make install

Instala las librerías necesarias para el manejo de los dispositivos de adquisición.

Una vez completados todos estos pasos, sólo se necesita reiniciar el sistema para que cargue los módulos necesarios.

Nuestro dispositivo USB-DUX es cargado automáticamente por los "scripts" de la utilidad "hotplug" cuando es conectado a un puerto USB.

Antes de realizar la programación en Comedi, se deben hacer las conexiones necesarias de las librerías compartidas con el comando.

#ldconfig /usr/local/lib

Después de todo este procedimiento se puede realizar la programación.

Si se quiere probar que el dispositivo USB-DUX a sido cargado correctamente, deberemos ver con el comando *dmesg* las líneas:

```
comedi_: usbdux0 has been successfully initialized
comedi0: usbdux: usb-device 0 is attached to comedi
comedi0: successfully atached to usbdux
```

Estas líneas pueden variar un poco dependiendo del sistema operativo.

ANEXO E
PROGRAMACIÓN COMEDI

En la programación Comedi hay una relación entre el controlador que ofrece el fabricante y el programa que hará el usuario.

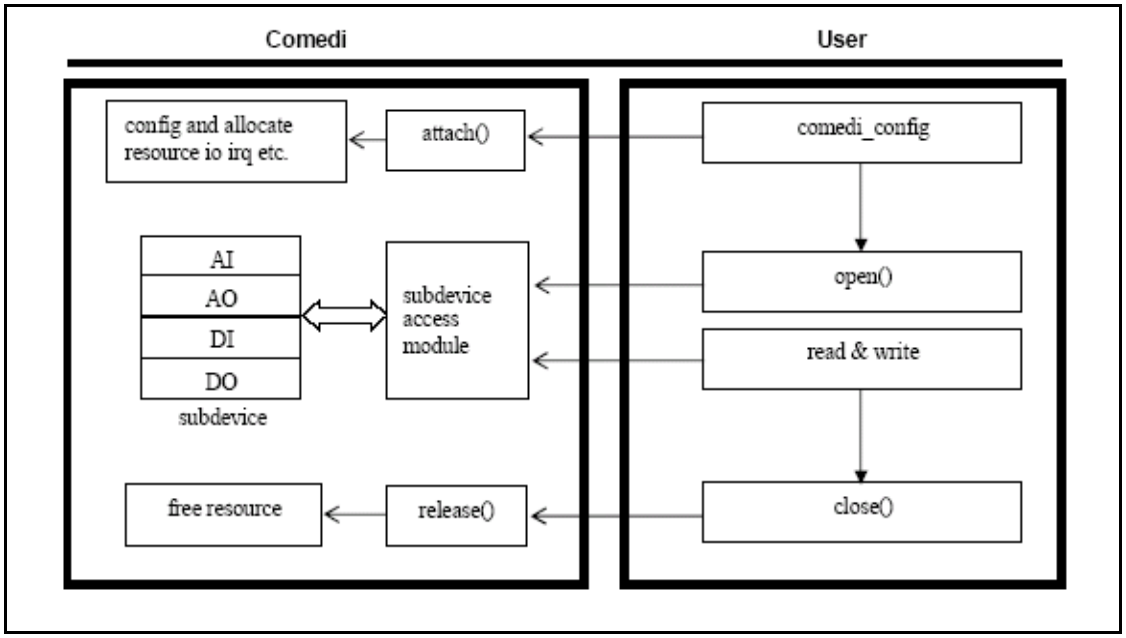

Figura E.1: Relación entre el controlador y el programa de usuario

La función “comedi_config” se ejecuta automáticamente al insertar el USB-DUX al puerto USB, esta función le da al USB-DUX la dirección, dentro del ordenador, para que el programa pueda acceder a él.

Las demás funciones del programa escrito por el usuario se ejecutan cuando son llamadas. La función “open”, aunque a la hora de programar tiene una sintaxis diferente, es la encargada de abrir el dispositivo para poder leer y escribir, y la función “close”, que le ocurre lo mismo que a la anterior, es la encargada de cerrar el dispositivo y liberarlo.

Todos los programas escritos en Comedi deben de seguir un diagrama de flujo general, como el que se muestra en la figura E.2, así se asegura que se accede correctamente al dispositivo a manejar.

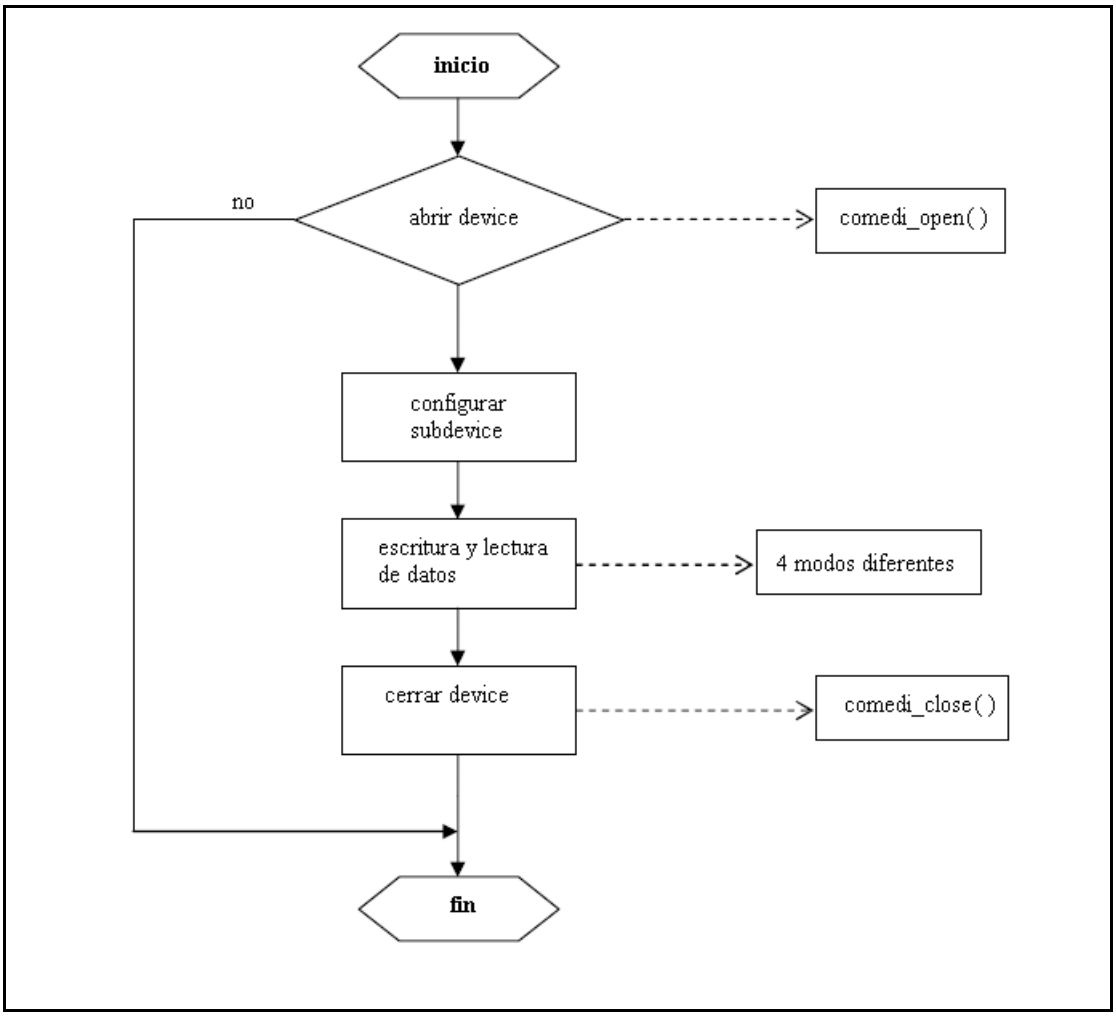

Figura E.2. Diagrama de flujo general

En Comedi, las diferentes funcionalidades de un dispositivo o device son llamadas subdevice; a la hora de programar hay que tener en cuenta el número de subdevice al que se quiere tener acceso. La siguiente tabla muestra los tipos de subdevices, el número con el que se designan los diferentes tipos, así como su descripción.

Nombre de subdevice	Tipo de subdevice	Descripción
COMEDI_SUBD_AI	1	Entrada analógica
COMEDI_SUBD_AO	2	Salida analógica
COMEDI_SUBD_DI	3	Entrada digital
COMEDI_SUBD_DO	4	Salida digital
COMEDI_SUBD_DIO	5	Entrada y salida digital
COMEDI_SUBD_COUNTER	6	Contador
COMEDI_SUBD_TIMER	7	Temporizador
COMEDI_SUBD_MEMORY	8	Memoria EEPROM,DPRAM
COMEDI_SUBD_CALIB	9	Calibración DAC
COMEDI_SUBD_PROC	10	DSP(digital signal processing)
COMEDI_SUBD_UNUSED	11	Sin usar
COMEDI_SUBD_FAKE	15	Dispositivo falso

Tabla E.1: Tipos de subdevices de Comedi

A la hora de programar en Comedi, se hace referencia a los subdevices a través de su número de subdevice dentro del dispositivo de adquisición y no de su nombre o su tipo. El subdevice COMEDI_SUBD_FAKE, se utiliza para el manejo de señales de interrupciones de entrada digitales. No todos los dispositivos tienen los mismos subdevices

Para este proyecto, el USB-DUX tiene 4 subdevices. El subdevice 0 es de tipo 1, las entradas analógicas. El subdevice 1 es de tipo 2, salida analógica. El subdevice 2 es de tipo 5, entrada y salida digital. El subdevice 3 es de tipo 6, contador.

Como se observa, el dispositivo de adquisición no dispone de todos los tipos de subdevices que puede controlar Comedi. Dispositivos diferentes al USB-DUX pueden tener otros tipos de subdevices, pero siempre serán acordes con los tipos mostrados en la tabla E.1.

E.1 FUNCIONES PARA ADQUISICIÓN SIMPLE

Es la forma más fácil de obtener los datos de un dispositivo, tanto digitales como analógicos.

Adquisición digital simple:

Comedi agrupa los canales digitales en un subdevice, que es un grupo de canales digitales que tiene las mismas características, como se puede observar en la tabla E.1.

Los bits individuales de subdevice de entrada y salida digital pueden ser leídos o escritos usando las siguientes funciones:

```
int comedi_dio_read(device,subdevice,channel,unsigned int *bit);
int comedi_dio_write(device,subdevice,channel,unsigned int bit);
```

El parámetro *device* es un puntero a un dispositivo Comedi abierto. Los parámetros *subdevice* y *channel* son número enteros positivos que indican que canal y que subdevice serán usados en la adquisición. El entero *bit* contiene el valor del bit adquirido.

La dirección de las líneas bidireccionales puede ser configurada usando la función

```
comedi_dio_config(device,subdevice,channel,unsigned int dir);
```

El parámetro *dir* puede ser también COMEDI_INPUT o COMEDI_OUTPUT.

También se pueden leer o escribir varios canales simultáneamente, para ello hay que utilizar la función:

```
comedi_dio_bitfield(device,subdevice,unsigned int write_mask, unsigned
                    int *bits);
```

Cada canal es asignado a un bit en el *write_mask*. Si un bit del *write_mask* es activado, el bit correspondiente en el dato **bits* será escrito en su salida digital

correspondiente. Ocurre lo mismo cuando el dato es leído. El valor del dato **bits* es indefinido y depende de cada dispositivo. Para nuestro dispositivo USB-DUX, el bit menos significativo del campo de bits es el canal 0; el bit más significativo es el canal 7.

Adquisición analógica simple:

Los canales analógicos en Comedi pueden producir valores, que son muestras de una señal analógica continua. Estos ejemplos son enteros con un rango que es normalmente, 8, 10,12,16 o 32 bits.

La siguiente función lee un valor de un canal y lo almacena en una variable especificada por el usuario, en este caso *data*.

```
int comedi_data_read(comedi_t device, unsigned int subdevice,usigned
                     int channel,unsigned int range,unsigned int
                     aref,lsampl_t *data);
```

Las variables nuevas que aparecen en esta función, son necesarias para la correcta conversión del dato; La variable *range* indica a la función en que rango de valores se encuentra el dato muestreado, y la variable *aref*, indica la referencia a masa que debe de tomar.

También existe una función para escribir un dato analógico en el canal especificado, esta función es:

```
int comedi_data_write(comedi_t *device, unsigned int subdevice,usigned
                      int channel,unsigned int range,unsigned int
                      aref,lsampl_t data);
```

Comedi proporciona también una función que nos permite obtener varios datos de un mismo canal:

```
int comedi_data_read_n(comedi_t *device, unsigned int
                       subdevice,usigned int channel,unsigned int
                       range,unsigned int aref,lsampl_t
                       *data,unsigned int n);
```

El número de muestras, *n*, esta limitado por Comedi a un número máximo de 100, pues la llamada a esta función bloquea al proceso llamador.

Otra posibilidad que ofrece Comedi es el retardo del comienzo de la adquisición un número determinado de nanosegundos:

```
int comedi_data_read_delayed(comedi_t *device, unsigned int
                             subdevice,usigned int channel,
                             unsigned int range, unsigned int aref,
                             lsampl_t *data,unsigned int nano_sec);
```

En los prototipos de estas funciones se han utilizado dos tipos de estructuras que no se han mencionado antes.

- **comedi_t:** Esta estructura de datos contiene toda la información que el programa debe conocer para un dispositivo de Comedi abierto. No es necesario llenar esta estructura manualmente, pues se rellena automáticamente cuando el dispositivo es abierto.
- **lsampl_t:** Esta estructura de datos representa un único dato. En muchas arquitecturas, no es más que un entero de 32 bits.

E.2 FUNCIONES PARA ADQUISICIÓN MÚLTIPLE

Las “instrucciones” son las funciones más flexibles que tiene Comedi. Son usadas para ejecutar una adquisición múltiple en un mismo canal. Una “lista de instrucciones” es una “instrucción” pero en diferentes canales. Ambos comandos son ejecutados de forma asíncrona, esto quiere decir que el proceso llamador se bloquea durante la adquisición.

Estructura de la instrucción:

Toda la información necesaria para ejecutar una “instrucción” es almacenada en una estructura de datos de tipo *comedi_insn*.

```
struct comedi_ins_struct{
  unsigned int insn;          //entero que determina el tipo de adquisición
```

```
    unsigned int n;             //numero de muestras
    lsampl_t *data;             //puntero a la variable que contiene los datos
    unsigned int subdev;        //"subdevice"
    unsigned int chanspec;      //especificaciones del canal
    unsigned int unused[3];
} comedi_insn;
```

Debido a la gran flexibilidad de la "instrucción", no se necesitan rellenar todos sus campos. Pero Comedi requiere que el campo donde se escribe el dato sea al menos de 1 byte de largo.

La variable *insn* dentro de la estructura de datos determina el tipo de adquisición.

- **INS_READ:** Lee el dato de un canal analógico.
- **INS_WRITE:** Escribe el dato en un canal analógico.
- **INS_BITS:** La instrucción debe leer o escribir un valor en varios canales de entrada o salida digital.
- **INS_GTOD:** La instrucción realiza un "Get Time Of Date".
- **INS_WAIT:** La instrucción bloquea el proceso durante un número determinado de nanosegundos**.**

Ejecutando la instrucción:

Una vez que la estructura ha sido rellenada la "instrucción" se puede ejecutar mediante la siguiente función:

```
int comedi_do_insn(comedi_t *device, comedi_insn *instruction);
```

Las "instrucciones" permiten ejecutar una "lista de instrucciones" en una sola llamada a la función:

```
int comedi_do_insnlist(comedi_t *device, comedi_insnlist *list);
```

El número de instrucciones en la lista esta limitado en Comedi, pues las "instrucciones" son ejecutadas de manera sincronía, y por lo tanto el proceso es bloqueado hasta que la lista de instrucciones haya terminado.

E.3 FUNCIONES PARA ADQUISICIONES CONSECUTIVAS

Los "comandos", o como los llama Comedi, "commands", es la función de adquisición más potente de todas las funciones de adquisición, pues permite hacer una secuencia de adquisición infinitamente para cualquier número de canales en cualquier orden y con señales de disparo, tanto externas como internas, para la adquisición.

Estructura del comando:

El "comando" se ejecuta de acuerdo con la información que se almacena en la estructura llamada "comedi_cmd"

```
struct comedi_cmd_struct {
    unsigned int subdev;    //dispositivo que se muestrea.
    unsigned int flag;      //codifica las posibles configuraciones de la
                            //ejecución del comando.

    unsigned int start_src;  // evento que hace empezar la adquisición.
    unsigend int start_arg; //parámetro que influye en ese comienzo.

    unsigned int scan_begin_src;  //evento que hace empezar un
                                  //muestreo particular.
    unsigned int scan_begin_arg;//parámetro que lo influye.

    unsigned int convert_src;     //evento para empezar una conversión
    unsigned int convert_arg;     //parámetro que lo influye.

    unsigned int scan_end_src;    //evento para terminar un muestreo.
    unsigned int scan_end_arg;    //parámetro que lo influye.

    unsigned int stop_src;        //evento para terminar la adquisición.
    unsigned int stop_arg;        //parámetro que lo influye.

    unsigned int *chanlist;       //puntero a la lista de canales
                                  //muestreados.
```

```
    unsigned int chanlis_len;      //número de canales a muestrear.

    sampl_t *data;                 //dirección del buffer de datos.
    unsigned int data_len;         //número de muestras.
} comedi_cmd;
```

Ejecutando un comando:

Un comando es ejecutado por la siguiente función de Comedi:

```
int comedi_command (comedi_t *device, comedi_cmd *command);
```

Rellenar la estructura puede ser algo complicado, por lo tanto, antes de lanzar el comando se recomienda hacer una prueba para saber si se ha rellenado la estructura correctamente. Comedi tiene una función para comprobar esto, simplemente de debe ejecutar la función "comedi_command_test( )" al menos una vez.

Eventos de disparo de un comando:

La estructura de comando tiene cinco tipos de eventos: empezar la adquisición, empezar el muestreo, empezar la conversión, para el muestreo y parar la adquisición. Cada evento puede estar dado por su propia señal, (la variable *_*src*) y cada señal tiene su propio argumento, (la variable *_*arg*). Por ejemplo, si se quiere especificar que la línea digital "3" actúe como señal de disparo, se deberá usar *src*=TRIG_EXT y *arg*=3.

El comienzo de una adquisición está controlado por el evento *start_src*. Las opciones disponibles son:

- **TRIG_NOW:** La adquisición empieza justo después de la llamada a la función "comedi_cmd". Solo funciona con la línea digital 0.
- **TRIG_FOLLOW:** La adquisición empieza cuando los datos son escritos en el buffer, (solo para dispositivos de salida).
- **TRIG_EXT:** Empieza cuando una señal de disparo se activa.
- **TRIG_INT:** Empieza cuando se activa una señal interna de Comedi, esta señal es producida por la instrucción INSN_INTTRIG.

El comienzo de cada muestreo está controlado por la variable llamada *scan_begin_src,* las opciones son:

- **TRIG_TIMER:** El muestreo empieza cada cierto tiempo. El tiempo viene definido por *convert_arg* en nanosegundos.
- **TRIG_FOLLOW:** El muestreo empieza inmediatamente después que otro muestreo termine.
- **TRIG_EXT:** El muestro comienza cuando una señal de disparo externa se activa.

El tiempo entre cada muestra en una instrucción de muestreo esta controlado por los campos *convert_src* y *convert_arg.*

- **TRIG_TIMER:** Las conversiones ocurren periódicamente. El tiempo entre conversiones es el especificado en la variable *convert_arg*, en nanosegundos.
- **TRIG_EXT:** La conversión empieza cuando una señal de disparo externa se activa.
- **TRIG_NOW:** Todas las conversiones de un muestreo ocurren simultáneamente.

El final de cada muestreo casi siempre se especifica usando TRIG_COUNT, con el argumento siendo el mismo que el número de canales en la variable *chanlist*.

El final de la adquisición está controlado por *stop_src* y *stop_arg.*

- **TRIG_COUNT:** para la adquisición después de terminar los muestreos.
- **TRIG_NONE:** genera adquisición continua, solo se para cuando se usa la función “comedi_cancel( )”

Su argumento está reservado y debe de ser puesto a 0.

Para determinar que señales de disparo soportan los “subdevices” se puede utilizar la función “comedi_get_cmd_src_mask( )”, pues no todos soportan todas la señales.

Los “flags” de un comando:

El campo llamado “flag” en la estructura de comando es usado para especificar el modo de funcionamiento en la adquisición.

- **TRIG_RT:** Utiliza el módulo de tiempo real de la Kernel, se debe de tener compilado Comedi con el soporte de tiempo real, sino esta opción no hará nada.
- **TRIG_WAKE_EOS:** "Ends Of Scans". Envía los datos al final de cada muestreo.
- **TRIG_ROUND_NEAREST:** Redondea al periodo de tiempo más cercano soportado.
- **TRIG_ROUND_DOWN:** Redondea hacia abajo.
- **TRIG_ROUND_UP:** Redondeo hacia arriba.
- **TRIG_DITHER:** Habilita la técnica de "dithering" sirve para suavizar el ruido que se genera en la discretización de la señal.
- **TRIG_DEGLITCH:** Habilita la técnica "deglitching", es otra técnica para suavizar el ruido producido por la discretización.
- **TRIG_WRITE:** Escribe en dispositivos bidireccionales.

Las funciones explicadas en este capítulo son las más importantes a la hora de la adquisición de datos.

Existen muchas más funciones de Comedi que realizan otras tareas, así como la obtención de información del dispositivo de adquisición, la conversión de datos, pasar de valores digitales a la magnitud física, en este caso voltios, y así un largo etcétera.

Para terminar, solamente decir, que Comedi, no se limita a nuestro dispositivo de adquisición, sino que cualquier dispositivo que el fabricante haya certificado con los controladores software de Comedi podrá funcionar sin problemas, siempre y cuando tengamos en cuenta, a la hora de programar, los distintos subdevices de que dispone.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Permanent-magnet and brushless DC motors.
Takashi Kenjo, Shigenobu Nagamori.
Oxford: Claredon Press, 1985.

Máquinas eléctricas.
Stephen J. Chapman.
Santa Fe de Bogotá: McGraw-Hill, 2000.

Instrumentación aplicada a la ingeniería.
Jesús Fraile Mora, Pedro García Gutiérrez.
Madrid: Servicio de Publicaciones, Revista de Obras Públicas, 1987.

C. Manual de referencia.
Herbert Schildt.
Madrid: Osborne / McGraw-Hill, 1999.

El Lenguaje de Programación C. Diseño e Implementación de Programas.
Félix García Carballeira, Jesús Carretero Pérez, Javier Fernández Muñoz, Alejandro Calderón Mateos.
Madrid: Pearson Educación / Prentice Hall, 2001

Diseño e implementación del manipulador móvil MANFRED
Ignacio Casillas Pérez
PFC (Ingeniería Industrial) presentado en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, 2005.

Estudio de la geometría de las secciones en los miembros del brazo robótico LWR-UC3M-1.
Sergio Moreno Gago.
PFC (Ingeniería Técnica Industrial: Electrónica Industrial) presentado en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, 2005.

The Control and Measurement Device Interface Handbook.
David Schleef, Frank Hess, Herman Bruyninckx.
http://www.comedi.org/documentation.htm, 2003.

Advantech Device Driver for Linux. User´s Manual.
Advantech Automation Corporation.
http://partner.advantech.com.tw/epartner/Files/Temp/user-manual.pdf

Installing Comedi Drivers. Kernel drivers for data adquisition card, instruction for instalation on Redhat 9 systems.
Steve D. Sharples.
http://optics.eee.nottingham.ac.uk/optics/documentation/hardware/installing_comedi.pdf

Archivos de instalación de Comedi.
Creados por los usuarios de Comedi.
Se pueden encontrar en los paquetes comedi.tar.gz o comedilib.tar.gz.
http://www.linux-usb-daq.co.uk/drivers2

Características USB-DUX.
http://www.linux-usb-daq.co.uk

Catálogo de la serie RBE(H).
Motores Kollmorgen.

Características. BE25A20 Series Brushless Servo Amplifier.
Advanced Motion Control.

Engineering notes. BE25A20 Series Brushless Servo Amplifier.
Advanced Motion Control.

Catálogo. Harmonic drive de la serie HFUC-2UH.
Harmonic Drive AG.

Engineering notes. Harmonic drive de la serie HFUC-2UH.
Harmonic Drive AG.

User Manual PMAC
Delta Tau.

Apuntes de la Universidad Carlos III de Madrid. Asignatura "Sistemas Informáticos en Tiempo Real".

Apuntes de la Universidad Carlos III de Madrid. Asignatura "Señales y Sistemas".

Manuales y tutoriales de Matlab.

Manuales y tutoriales de Gnuplot.

Página Web:
http://www.uc3m.es/uc3m/dpto/IN/dpin04/manfred_web.html#Software.

Página Web:
http://www.tmt.ugal.ro/crios/Support/ANPT/Curs/sdyn/s6/s6fmathm/s6fmathm.html.

PÁGINAS WEB DE INTERÉS

http://www.comedi.org/download.html
Se puede encontrar la última versión de los paquetes Comedi y Comedilib.

www.swig.org
Se puede encontrar el programa que permite compilar el paquete Comedilib.